IMPROVING PROJECT MANAGEMENT PLANNING AND CONTROL IN SERVICE OPERATIONS ENVIRONMENT

Mohamed Al-Kaabi

A thesis submitted in partial fulfilment of the requirement of De Montfort University for the Degree of Doctor of Philosophy

August 2011

Abstract

Projects have evidently become the core activity in most companies and organisations where they are investing significant amount of resources in different types of projects as building new services, process improvement, etc. This research has focused on service sector in attempt to improve project management planning and control activities.

The research is concerned with improving the planning and control of software development projects. Existing software development models are analysed and their best practices identified and these have been used to build the proposed model in this research. The research extended the existing planning and control approaches by considering uncertainty in customer requirements, resource flexibility and risks level variability. In considering these issues, the research has adopted lean principles for planning and control software development projects.

A novel approach introduced within this research through the integration of simulation modelling techniques with Taguchi analysis to investigate 'what if' project scenarios. Such scenarios reflect the different combinations of the factors affecting project completion time and deliverables. In addition, the research has adopted the concept of Quality Function Deployment (QFD) to develop an automated Operations Project Management Deployment (OPMD) model. The model acts as an iterative manner uses 'what if' scenario performance outputs to identify constraints that may affect the completion of a certain task or phase. Any changes made during the project phases will then automatically update the performance metrics for each software development phases. In addition, optimisation routines have been developed that can be used to provide management response and to react to the different levels of uncertainty.

Therefore, this research has looked at providing a comprehensive and visual overview of important project tasks i.e. progress, scheduled work, different resources, deliverables and completion that will make it easier for project members to communicate with each other to reach consensus on goals, status and required changes. Risk is important aspect that has been included in the model as well to avoid failure. The research emphasised on customer involvement, top management involvement as well as team members to be among the operational factors that escalate variability levels

and effect project completion time and deliverables. Therefore, commitment from everyone can improve chances of success. Although the role of different project management techniques to implement projects successfully has been widely established in areas such as the planning and control of time, cost and quality; still, the distinction between the project and project management is less than precise and a little was done in investigating different levels of uncertainty and risk levels that may occur during different project phase.

Acknowledgments

Praise to Allah who has guided me through and has given me the strength of the determination to carry out this work.

Reaching the finishing point of this journey, I deeply realise that I was not walking it alone; however I was surrounded by caring people who encouraged, exhorted, and supported me to pursue the race. I would like to take this space to acknowledge them.

First I owe my profound gratitude to my family here in the UK and back in United Arab Emirates for their unending patience, support, encouragement, and love.

It would have been next to impossible to finish this research without help, guidance, and endless support from my supervisors (Dr Riham Khalil and Prof. David Stockton) who have provided invaluable encouragements, advices, suggestions, and comments. They made me realise that impossible is doable. Thank you from the deepest point of my heart for all the effort you gave me.

A special thought goes to my father for his continuous encouragement and support to achieve goals, to my mom for her continuous prayer and care, for my brother and sister for their love.

I also want to thank my wife, who carried a lot of heavy stress on her shoulder to allow me having a clear mind in doing this thesis and for her love. I wish to thank my two daughters Meena, Mahra and my son, Sultan for their smile that showed me the meaning of life.

With a deep sense of gratitude, I wish to thank the government of United Arab Emirates for allowing me the chance to persuade my goal and supporting me in all ways to reach the end in this journey. In addition, I would like to thank the IT department employees in ADNOC Company for their time in providing the needed data to build my simulation model and validating parts of the research proposed steps.

Finally it is pleasure to thank my second and special family at the lean research group/ centre for manufacturing for their support, suggestions and care.

Declaration

I declare that the work described within this thesis was originally undertaken by me, (Mohamed Al-Kaabi) between the dates of registration for the degree of Doctor of Philosophy at De Montfort University, April 2008 to August 2011.

List of Tables

Chapter 2 Operations Planning and Control

Chapter 3 Services Operations

Chapter 4 Research Methodology

Chapter 5 Experimental Results

Chapter 6 Discussion

List of Figures

Chapter 3 Service Operations

Chapter 4 Research Methodology

Chapter 5 Experimental Results

Chapter 1: Introduction

1.1 Background

In today's global market, managing service projects is a crucial task due to the many entities, dynamic changes and unpredictable relationships between them. Although most of the service problems are identified, they are poorly acknowledged (Petter 2008; Rautiainen et al. 2003).

The difficulty faced by project management team in dealing with projects e.g. software project, arises from the uncertainty in planning the deliverables and controlling the progress of the project, fragility and clarity in requirements, and dynamisms characteristics in such type (Roy, 2007). Having a successful project needs a management process that balances between activities that add values, and the ones that do not add value but are essential, and eliminates activities that neither adds values nor core for other processes.

The research here used software development project to be a case study to develop a quick response model that can plan and manage any uncertainty that may occur among project phases. Improving software project management took different forms in literature, e.g. cost estimations, reusability of code, enhancing compilers, building new programming languages, etc. However, much less attention has been paid to ways that can improve project management efficiency and reduce overruns (Zhang et al. 2007). Consequently, factors that increase/decrease stability of the software project during execution (development) time are important to be identified through studying the operations of such environment (Krasna and Rozman 1996).

1.2 Planning and Control as a Management Paradigm to Services

Literature reviews have shown that project management team has diverse responsibilities but their most significant tasks are planning, estimating, scheduling, and executing the plan. These activities are continuous and iterative throughout the course of the project rather than to be seen as too rigid or only a pre-phase to production (Perminova et al. 2008). Successful plan considers (Wright et al. 2009):

a. the high level of uncertainty among project phases,
b. changes in customer demands,
c. setting up project deliverables in relation with project complexity and customer's needs,
d. project scheduling that involves tasks duration and resources (materials, equipment, people) allocations, and
e. risk management to identify, assess and mitigate their factors.

Improvements may involve adopting different development lifecycles and techniques from other development fields than service industry in enhancing operations management (McBride, 2008). Accordingly, this research has adopted lean thinking that would help in understanding the process, highlighting the different problems that may occur and determines the added and non-added value activities. In addition, lean also helps to investigate the level of variability within these activities and improve the outcome/performance of the project (Khalil et al. 2006).

Similarly, risk management is an important element to improve project planning. Decisions can be made to avoid, mitigate, or accept some levels of risk. Identifying risk factors can help root their causes and tackle them at the source.

Quality Function Deployment (QFD) is one of lean tools that help in increasing efforts focus and process coherent as the voice of customer is captured and translated into service. The research here is aiming to adopt the concept of QFD to develop a proposed method that help in planning and control project activities.

1.3 Process Orientation Improvement

Projects are not limited to a specific boundary in any organisation as they are considered to be a horizontal process rather than vertical (departmental). Throughout history, most of the improvement attempts approached by industrial activities and organisations are mainly intense on single and isolated tasks, functions, departments, or what is known to be vertical improvements (Ljungberg, 2002). However, less attention is paid to improve the horizontal stream, i.e. cross-functional process in the organisation that views the organisation as a whole (Narasimhan and Jayaram 1998). The research here will consider the latter ideology as the base to manage, plan and control the service operations.

In considering the process orientation, there is almost no boundary limits for the area of responsibility. However, there is a need for process owner (e.g. leadership and responsibility to be adopted as suggested by one of the proposed plan and control model characteristic in Section 2.4) who is dedicated to the management and development of

the entire process. Multi-skills is another necessity in process orientated organisation as individuals can fill the empty or needed space as queues start to form which enhances flexibility. The research here will investigate the possibility of achieving flexibility and manage different resources among the projects phases.

1.4 Lean Management

In modern days, it is not the size/scale of business that creates competitive advantage neither the inflexible systems of goods and services where rapid response counts in weeks and months rather than days or less. Such systems can be considered as liability instead of asset (Lee-Mortimer 2006; Proctor and Doukakis 2003; Parry et al. 2010; Hines et al. 2006; Toni and Tonchia 1996).

Thus a new philosophy pioneered by Toyota in the late 1980's named lean came to address this critical issue. Lean management entered the management field widely as it promises to cut costs and improve quality. Moreover, lean tends to match supply with customer demand through stabilising operations and capturing customer requirements and needs, thereby achieve continuous and iterative improvements.

In a lean organisation, it is the consumer who controls the pace, specifies goods and services to be produced, rather than reviewing historical data or following arbitrary targets. In the old mass paradigm, the producers tend to limit their services with a set of goals that accept some number of defects, certain size of inventory, and a very narrow range of standardised products and services. However, with lean management, the firms aim to achieve perfections in their services, declining in their overall costs, zero defects,

very low levels of WIP and wide range of product and services variety (Riezebos et al. 2009).

1.5 The Aims and Objectives of Research

The research seeks to develop a planning and control model, entitled 'Operations Project Management Deployment (OPMD)'. The developed method can be used as a quick responsiveness tool that will visualise the different 'what if' scenarios and how the team member can react to any uncertainty or if risks occur that can affect project deliverables or milestones. In other words, it can identify bottlenecks, different constraints, and move resources accordingly when and where they are needed.

The research aims will be achieved through implementing the following objectives:

a. Capturing customer demands to be the goals of the project, i.e. indicators of success when they are achieved.
b. Understanding software process by identifying the factors/variables that can affect project completion time and its deliverables.
c. Analysing existing software models to map down the software project and adopt best practices within the proposed model.
d. Adopt quick response to uncertainty or change by optimising critical constraints to project outcome.
e. Avoid failure by raising risk levels when outcome are not matching customer demands.
f. Achieve flexibility by moving operators from downstream to upstream and vice versa i.e. when and where they are needed.

This model will be built according to lean thinking and guidance through delivering only value to end customer and improving project performance. Moreover, the research here will suggest a proposed standard steps to plan and control projects.

1.6 Structure of the Thesis

The thesis starts in **chapter 1** with a brief introduction about the difficulties that faces any project management team to plan, and control a service project or any project with high dynamic characteristics. Attention to process improvement rather than departmental is explained then. Chapter 1 concludes by stating the aims and objectives of the current research.

Chapter 2 introduces briefly the concept of operations management and their basic types i.e. manufacturing and service systems. The chapter then moves to the core subject of this research, planning and control. It defines the concept and illustrates the different types of planning and control processes. Chapter 2 next explains the characteristics of the plan and control model. Such characteristics are being utilised as a guideline for the enhancement process of planning and control. How other researchers dealt with planning and control issues and the different models, approaches, and methods that have been developed to plan and control service projects and their limitations are shown next.

Chapter 2 then introduces lean concept as it is the engine to enhance the planning and control process. It explains the five principles of lean, defines what is value and waste, and then states the seven wastes as defined by lean. By adopting the Toyota 14 ways of

management, the research explained the steps how the operations efficiency of a service system can be enhanced.

Chapter 3 discusses in more details the different aspects of service operations. It starts by identifying the specific characteristics of service operations system. Those characteristics help to realise the difficulty and weak points of previous planning and control models in dealing with such type of projects. The software process is being explained with a breakdown structure of the various activities for each phase of the process. Then the chapter critically reviews the literature on different project management tools, models, and approaches.

The chapter next critically explains the weaknesses and strengths of existing software development models to deploy their best practices in a standard process. Chapter 3 then illustrates in details the lean software development model (LSD), its principles, practices, and the seven waste in such environment.

The chapter proceeds by covering the concept of project management as it is a major aspect in planning and control. The different project management models and tools are critically been considered with attention to software project. Therefore, Quality Function Deployment is being introduced next as one of lean's project management tools that the research can make use of its concept. Then the chapter proceeded by briefly covering the literature of the QFD enhancement to suite service operations. At that point, the chapter critically reviews the different models to manage customer requirements/demands through the use of QFD.

Through literature reviews, different structures and techniques of capturing customer demands do exist. The research here has investigated their use, strength and weaknesses. Finally, the chapter concludes with critical analysis to the different methods of risk management and assessment in managing planning and control in service operations environment.

Chapter 4 describes the research undertaken to develop the research methodology. Such methodology combined both quantitative and qualitative types. Taguchi's Orthogonal Arrays is applied as Design-of-Experiment. Based on the aims and objectives of the research, the experimental design is chosen to investigate expected improvements to achieve such aims. In that, Regression analysis is used along with Genetic Algorithm optimisation to provide optimum solution to the critical software factors that affect project completion time and its throughput.

Chapter 5 reports the results of the simulation experiments after running the required number of experiments according to Taguchi's Orthogonal Arrays. Moreover, it draws attentions to the key factors playing major effect in raising or reducing variability within the lifecycle of a service project, in this case software project, and affect its completion time.

Chapter 6 discusses the results and major findings obtained in chapter 5. It analyses the relationships between the ten different operational factors that are identified in chapter 3 (i.e. through literature reviews, case study, and interviews), and their effects on the performance measurements.

The chapter proceed by listing the best practices that have been included in the OPMD model from previous mentioned software development models in chapter 3. The chapter then discusses the results and presents the research proposed steps for planning and control projects.

In addition, the chapter highlights three major points related to the OPMD:

i. The differences between OPMD and other planning models including QFD.
ii. The use of OPMD in new, already existing projects, and multi projects.
iii. The use of OPMD with existing plan and control models.
iv. The use of OPMD with existing manufacturing Scheduling.

Chapter 7 sketches the conclusion of the research and highlights the contributions of this research to knowledge.

Chapter 8 lays the ground for further research and investigation as it describes future work to extend the use of OPMD to include cost considerations, and human resources rather than activities or processes.

Chapter 2: Operations Planning and Control

2.1 Introduction

Literatures and interviews have shown that uncertainty in terms of planning with regard to any project is high initially. Therefore, project success in achieving the intended goals depends on having a clear vision of the factors that can increase the degree of uncertainty and risks associated with them. Divr and Lechler (2004) emphasised the importance and positive impact of using formal planning approaches with regard to the project success. It is considered that a carefully created plan is the foundation on which project success is built. Moreover, the plan helps in keeping participants updated and engaged (Hartman and Ashrafi 2004).

This chapter starts by defining operations management and presenting the two main operation types. The definition of planning and control is then illustrated. Characteristics of the proposed planning and control model are identified and presented next where they are used in developing the research methodology. Then a review of the different plan and control models used by researchers and organisations to manage services operations is critically illustrated. An introduction to lean thinking and its applicability in service operations are illustrated next. Toyota 14 ways are adopted and presented at last to provide an improvement framework to planning activities.

2.2 Operations Management Definitions

Lee and Schniederjans (1994) defined operations management as "T*he study of concepts, procedures, and technologies used by managers, administrators and employees in the operation of all organisations*".

Meanwhile, Meredith (1992) had defined operations as "*The process of transforming inputs into useful outputs and thereby adding value to some entity; this constitutes the primary function of virtually every organisation*".

In summary, operations management is a discipline that is concerned with activities that produce value through the management of human, technology and system resources (Greasley 2006; Stevenson 2005; Silver 2004; Lee and Schniederjans 1994). It dates back to the Industrial Revolution starting in the 1770s (Gantt 1916; Taylor 1911; Erlang 1909). The comprehensive core tasks of operations management include planning, controlling, directing, organising, staffing and motivating (Bertrand and Fransoo 2002; Meredith 1992).

2.2.1 Types of Operations Management

According to Meredith (1992), Lee and Schniederjans (1994) and Greasley (2006), different classification of operations exist e.g. product size, type of human resources or equipment used, etc. The research here has adopted the one that classifies operations according to its final output or product, either goods or services. Accordingly, operations are of two types;

a. Manufacturing Systems

Manufacturing is the process of converting raw materials into a variety of products according to customer demands (Kalpakjian, 1995). The outputs of the transformation of manufacturing systems are tangible goods and products such as textbooks, eye glasses, automobiles, canned food, television sets, etc. (Lee and Schniederjans 1994).

b. Service Systems

According to Looy et al. (2003), services can be defined as activities that are of a more or less intangible nature. Such intangibility can be seen in the interaction between the customer and the service provider as a solution to the customer's needs. Service is a combination of experience and outcome which together are referred to as service package or product (Aurich et al. 2010; Johnston and Clark 2008).

Usually the outputs of such system are intangible products such as travel advice from the Thomas Cook travel agency, health care, transportation, information, hotel lodging, etc. (Meredith, 1992). However, there is no distinct line separating services from manufacturing, as the transition occurs gradually (Bicheno 2008; Greasley 2006; Ruch et al. 1992).

The research here will study and analyse the second type of operations system, service operations. It will investigate the different types of variability that may affect project success. Attention will be paid more to planning and control activities for different size of projects. Software project is used in this research as a case study to apply the analysis through as it reflects the specific characteristics of service sector as will be shown in Section 3.2.

2.3 Planning and Control

In planning and control processes, the main objective is to set up clear steps that transform the inputs (i.e. mostly obtained through customer demands and requirements in a step named here the specification phase) into outputs to fit that demand. In this

manner, Lee and Lee (2006) have described software project planning as the representation of a project network of stages and activities, which contain lower level detailed tasks, and precedence restrictions.

2.3.1 Planning Process

According to literature reviews**, Planning** is the process of setting goals, reasons for choosing them and actions to accomplish them, with enough details in regard to schedules, costs and other factors that affect the execution of such goals (Amescua et al. 2004; Lewis 2001; Johnston and Brennan 1996). It seeks to exercise a favourable influence over future events to achieve the intended goals (Head, 1984).

According to Head (1984), the planning process consists of three main steps:

a. **Formulation**: a draft plan is prepared that contains a set of proposed objectives.
b. **Review**: to ensure that information resources are optimally allocated, and during which a great deal of emphasis is placed on objectives that are aligned with the business goals of the firm or organisation.
c. **Tracking**: i.e. monitoring chosen Key Performance Indicators (KPI) against agreed-upon objectives as will be shown in Section 4.4 Step 4.

The initial activities of any project play a major influence on final project outcomes. Their impact is at its highest at the start, and decrease rapidly throughout the progress of a project. Moreover, cost of making changes is lowest during the early stages and increases rapidly through the project completion. For example, changing customer requirements at an early stage is easier than when the company reaches the design or

implementation stages. Accordingly, a well-defined plan can reduce risks, failure, and cost of the project (Lewis, 2001).

2.3.2 Types of planning activities

Planning in most organisations is of three types according to their breadth or time frame (Hans et al. 2007; O'Regan and Ghobadian 2002; Sutton 1996; Lee and Schniederjans 1994):

a. **Strategic planning**: i.e. concerns with long-term goals and involving upper management activity.
b. **Tactical planning**: i.e. concerns with medium-term goals.
c. **Operational planning**: i.e. concerns with shorter-term goals such as daily activities on the shop floor.

The previous categorisations deal with different types of decisions, operational and managerial levels, time horizons, levels of detail, and modelling assumptions. The focus of the research here is on operational i.e. completion of activities/tasks within every phase rather than on strategic and tactical planning. Focusing on daily and short term activities helps to understand the constraints of flow within project management and types of wastes. Moreover, it provides more control over available resources and project activities rather than long vision plans.

The plan can be either proactive, i.e. it tends to reduce the consequences or impact of uncertainties before project start date (e.g. allocation sufficient slack time, resources, outsourcing); or reactive that absorbs disturbance of un-avoided events such as sickness, etc. (Hans et al. 2007; Philpott et al. 2004). Combining both types in a unified model

will significantly help management team in solving issues related to project activities planning.

2.3.3 Control Process and its Types

Control can be defined as the process of guiding the system toward a pre-determined standard or goal through the comparison of actual performance with planned measurements (Bonner 2005; Abdel-Hamid et al. 1993). Collyer and Warren (2009) defined control to be the exercise of managing resources (e.g. human, equipment, tools, etc.) with continuous comparison to planned performance and taking steps to correct any deviation as means of allowing project to achieve its objectives.

McBride (2008) has categorised control activity into four types;

a. **output**, e.g. budget, schedule, and functionality,
b. **behavioural**, e.g. plan, formal process,
c. **clan**, e.g. exchange developer, site visits by the project manager, and informal communications, and
d. **input control**, e.g. project manager selection, and team selection.

Control in most cases is accompanied by monitoring activities. McBride (2008) has defined project **monitoring** as the activities of gathering information and measurements of the project in order to determine its current state. Monitoring is of four types:

a. **automatic**, e.g. workflow, configuration management, release management,
b. **formal**, e.g. product review, schedule and milestone tracking, team meetings, customer and management review,

c. **ad hoc**, phase end review, drill down inquiry, and
d. **informal**, conversations with team or stakeholders or customer (McBride, 2008).

Both planning and control are non-separable activities in managerial terms (Ebert 1999; Taylor 1999). They are about matching customer burdens to operational capacity (Greasley, 2006). Therefore, planning, control, and monitoring are the activities that will be integrated as means of offering a better judgement on the part of the management team in order to judge the success of the project.

The mechanism of control this research is applying in the proposed method will be through comparison between the readings of the identified performance measurements and customer demands. Therefore, this research has pooled McBride's (2008) categorisation (integrated the four types in the proposed method) of control as a structure of sustaining success in project operations.

2.4 Characteristics of Planning and Control Model

Identifying certain characteristics in the processing environment helps standardise development and clears the vision in terms of task management improvement. The following characteristics will be deployed as guideline indicators for improving the operations management plan and control any service environment project. They will be blueprinted in the research proposed method 'Operations Project Management Deployment' as will be shown in later chapters. These characteristics are as listed in Table 2.1.

- **Formal Process:** this includes standard tasks within the software lifecycle process. Otherwise, a lack of formality leads to bad quality and time consumption. Applying formality has been motivated by the expectation that performing appropriate mathematical analyses can contribute to the reliability and robustness of the product in a similar manner to other engineering disciplines (Hughes and Cotterell 2006; Gould 2006; Liu et al. 2008).

- **Even Specifications:** it is necessary to design what customers need at the time (Poppendieck and Poppendieck 2007). Adding extra specifications to the modules of the project can lead to over-tasking which is expensive in terms of effort and time.

- **Flow of Information:** information availability in each phase has to be sustained. This includes accuracy, easy access and being understood by the team members (Birk 2002).

- **Allocation of Tasks** this is based on their complexity and scope. These tasks are assigned to skilled teams with timing and action details.

- **Knowledge:** theoretical and practical understanding of programming skills, managing, testing, documenting and core knowledge of the software field.

- **Knowledge Management:** ability to confine, store, sort, recover, and manipulate software product information and its development attributes in an inclusive perspective (Gould, 2006). The lack of efficient knowledge management contributes to a significant increase in project lead time, value delivered, and effort wasted (Ebert and Man 2008).

- **Reliability:** this includes an understanding of different problems and failures that may occur within the development process. This will help the process to operate correctly, completely and consistently for a specified period of time (Gong et al. 1998; Bernstein 1994).

- **Flexibility:** this refers to the ease of the framework to respond to uncertainty in business conditions during project execution while sustaining or increasing the delivery value (Schonsleben, 2007). Providing different choices can enhance flexibility to accommodate inevitable changes (Hartman and Ashrafi 2004).

- **Risk Management Analysis:** risk consideration prior and during project execution is important to avoid failure. Adopting risk mechanism will highlight the critical path within the project activities.

- **Decision Making:** the model can help the project management team to choose the proper actions to be taken in relation with the identified project goals. This can be undertaken in the different phases of the software project.

These decisions will determine the improvements and tasks involved in completing the project.

- **Leadership and Responsibility:** where the model encourages the involvement of all team members (including customer and top management) in developing the plan and sharing responsibility. Moreover, assigning a member who has leadership, technical and managerial experience to be champion. This helps the flow of the horizontal value stream through the vertical functional departments of the organisation (Womack and Jones 2003).

- **Visual Management:** it is the ease in visualising the progress and key performance indicators of the project which helps in understanding the daily tasks, responds quickly to problems and obstacles, and eliminates communication overhead waste (Gould, 2006).

- **Coordinated and connected:** the different phases within the process are connected. A change in one phase, due to internal or external factors (requirements changes, new technology appearance, etc.), is reflected in others (Gould, 2006).

Table 2.1: Characteristics of Operations Management Plan and Control Model

In contrast, through the literature review, some specific characteristics or constraints have been identified that add more complexity and high-dynamics to the software project. The following characteristics are part of the software service operations that will be discussed in details in section 3.2. Studying these characteristics defines the boundaries of the software operations. These characteristics include:

a. **Invisibility:** software projects are purely intellectual, intangible products. They are devoid of physical characteristics, i.e. mass, weight, colour, and cannot be experienced separately from the operating system. (Hughes and Cotterell 2006; Ben-Menachem 2008). Therefore, management team is faced with additional pressure when it comes to planning, monitoring and controlling the progress of the project.

b. **Fragility of requirements and ease of system modification:** the ease of modifying, deleting and copying software systems push customers to continue changing what they ask for. High customer contact in this type of operation increases the volatility of requirements (Hughes and Cotterell 2006).

2.5 Existing Methods for Planning and Control

The literature offers various methods of planning; some researchers tend to use classical charts, whereas others argue for the use of systematic planning, due to its simplicity in use for non-specialised employees (Luo et al. 2010; Wagner et al. 2009; Ljungberg 2002; Lehtonen et al. 1999).

Applying benchmarking through maturity analysis (Capability Maturity Model, Organisational Project Management Maturity Model, etc.) in order to plan a project has been studied by Zwikael and Globerson (2006). Their study aimed to identify the best practices for planning and executing a project, and then employ it as a benchmark for improving project planning in other industries. However, their analysis was based on four industries and this has limited the number of identified best practices. In addition, benchmarking activity is only one step in the planning process; it should be combined or aligned with risk analysis and other planning activities.

Hans et al. (2007) has developed, through a survey and two case studies, a generic hierarchical planning and control framework that supports multi-project planning. In a project with high complexity and uncertainty rates, this framework enables the selections of the appropriate methods of planning based on organisational

characteristics. Their framework is specified to multi-project rather than a single project.

Jonsson and Wohlin (2006) developed a generic checklist with three axes, involving process, domain and roles. Their approach reduces the number of changes throughout the lifecycle of software project. Their analysis has highlighted a core operation factor, i.e. the volatility of requirements, which the research here has investigated its effect. However, more factors play a major role in the planning process and need to be identified.

Kececi and Modarres (1998) developed a new software development lifecycle model that is based on the Goal Tree Success Tree and Master Logic Diagram (GTST-MLD). This model helped to visualise any missing/incomplete requirements, and decomposes the complex software process into independent modules. Nevertheless, risk analysis was not examined to enhance this model. Whereas Fragniere et al. (2010) applied aggregate planning model (APM) to services rather than to goods by optimally planning the qualified workforce capacity constraint.

Greer and Ruhe (2004) developed a method (entitled EVOLVE) to plan and control an iterative lifecycle development. This model had adopted Genetic Algorithm to promote the best solutions. EVOLVE helped prioritise the different functionality, solved the conflict between the different stakeholders, and balanced the load between needed requirements and available resources. However, this approach was limited to only provide a solution for non-convex problems.

Hartman and Ashrafi (2004) have developed a unified generic planning framework 'SMART'. It has been developed through empirical data analysis using data collected in more than 8 years, i.e. critical success factors, key performance indicators, causes of failure, and project drivers (priorities). The name stands for '**S**pecific, **M**easurable, **A**ctions, **R**esources, and **T**ime' that represent the columns of the SMART plan template. SMART consists of SMART Breakdown Structure (i.e. project mission, identifies key stakeholders, groups the expectations, and identifies the tangible deliverables), priority triangle that uses 'Three Key Questions' which include:

a. What is the final deliverable for this project?
b. Why is everyone praising this project?
c. Who will decide the answers and outcomes of the two previous questions?, and RACI (**R**esponsibility, **A**ction, **C**oordination, and **I**nformation) chart.

This framework is very similar to what the research is intended to develop. However, more attention is needed to link the performance metrics to both customer demands and operational variables. This can be accomplished through the use of simulation as a control tool.

Collyer and Warren (2009) suggested the need to manipulate the project environment from dynamic to static (e.g. freezing objectives and design, delaying the adoption of new technologies), scope control, applying proper lifecycle strategies (e.g. waterfall, spiral, etc. which will be covered in more detail in Section 3.4), scope & management control (e.g. input, behaviour, and output control), leadership style, controlled experimentation, and categorisation.

Other researchers have suggested the need to shift from the classical planning approaches to milestone planning. Thus, milestone planning is more of a result-oriented approach than an activity-oriented one, where a milestone is a result to be attained (Andersen et al. 1995; Turner 1993).

Reviewing the previous literature with regard to planning and control tools, approaches, methods, and frameworks has highlighted some of their limitations or constraints. Such limitations have prevented any major/significant improvements in the planning field or in increasing the success of projects. They are generic processes and practices that fail to address some of the technical, commercial, and environmental issues. Less emphasis has been on the front end in comparison with classical issues (time, cost, scope, and quality). There have been inadequate links between the three entities of any project;

a. project performance,
b. success factors (operational factors), and
c. customer demands.

Furthermore, there has been an inability to capture the differences in terms of stakeholders' perceptions towards final goals/project requirements. Therefore, the current research will focus on the core causes of such a lack which will include customer demands, operational factors, performance metrics, and risk factors.

Lately attention is given to include sustainability in consideration when comes to develop or enhance planning tools. It is about considering the long run of the operations with a careful analysis to its impact on future of organisation and customers, meeting

the current essentials without conceding the needs of future generations i.e. how to improve the current project operations with attention to avoid negative impact on future projects strategy or goals. To ensure high levels of sustainability within operations activities, Tahir and Darton (2010) had suggested measuring the performance with an appropriate set of indicators that balance between resource efficiency and fairness in benefits. The research here has a generic standard set of performance measurements as will be presented in Section 4.4 step 4 that consider efficiency. In addition, such PMs can be modified and added to them in order to enhance the sustainability of operations activities.

2.6 Definition of Lean

Lean was coined by researchers (Mehta et al. 2008; Gould 2006; Browning 2000; Raman 1998) as finishing the tasks on time while minimising the non-value-added activities and being flexible to change. Lean is about maximizing value (Browning, 2000). It accelerates the process speed through analysing flow, identifying causes of delays, and quantifying the costs of complexity (Sutton, 2008).

Lean enterprises originated in Japan after World War II within the Toyota Company. It was pioneered by Taiichi Ohno (1912-1990). Initially, this philosophy was called Toyota Production System (TPS). However, it should not be forgotten that Henry Ford had already begun applying parts of Lean principles in some of his factories as early as the 1920s (Kilpatrick 2003; Womack, Jones and Ross 1990).

At the start, lean principles were applied to manufacturing and production operation sectors e.g. automotive, aerospace industries (Hicks 2007; Ranky 2007). However, through literature reviews, lean principles proved their success in the services operation sectors e.g. health care, information management, food retailers, etc. (Pedler and Abbott 2008). Lean tools have been successfully implemented with minimal investment cost in pure service context, e.g. in financial services call centre (Piercy and Rich 2009), full-service life insurance company (Swank, 2003), Toyota car dealer (Whereas Kosuge et al. 2009), etc. Thus, the applicability of the lean philosophy in enhancing operations relating to software projects have been noted (Aoyama, 1996).

2.6.1 Lean's Five Principles

The following five principles are core to the implementation of lean production (Abdulmalek and Rajgopal 2007):

i. **Value in the voice of customers:** value can be achieved through the ultimate customer description that pays back what has been spent through operations. Hence, it is important to capture exactly what the customer wants or needs, and to transform it into production or operational activities (Khalil et al. 2010).

ii. **Mapping the value stream**: through visualising the whole stream of the product and eliminating waste, i.e. non-value added items (Hicks 2007; Morien 2005). Attention paid to the customer or service provided rather than on the viewpoint of department or vertical boarders of the organisation.

iii. **Implementing the flow process**: once waste has been eliminated, let adding value process flows and gains accurate pace (Raman, 1998). There should be

no impedances in terms of queues or inventory shortages, any extra movements or transport.

iv. **Implementing the pull system**: the downstream pulls from the upper stream based on the demand of customers (Raman 1998; Bowen and Youngdahl 1998). Pull in services is a fast reaction and answer to customer's rate of demand (Bicheno, 2008).

v. **Perfection**: Always maintains the search for better performance with reduction in cost and time. Quality is enhanced and increased through the continuous improvements and elimination of undiscovered waste (Khalil et al. 2010).

2.6.2 Value and Waste

Waste, 'Muda' in the Japanese term, is considered by lean as any activity that does not add value to the end customer. Understanding waste is critical to achieve a successful implementation of the lean approach in any business. The Lean approach has led to the identification of the seven wastes that can occur in mainly any operational industry, whether service or manufacturing. Muda in production has been categorised into seven types (Womack and Jones 2003; Bicheno 2000) as follows:

1. **Overproduction**: is the most serious type of waste according to Ohno. It is the major cause of other wastes and problems that arise in operations management (Womack and Jones 2003). Overproduction is about making or producing more products/services than needed by the market or by the customer, too early or ahead of time, just in case to be needed (Bicheno, 2000).

2. **Waiting**: arises when the downstream process is waiting for upstream activities to be completed (Hicks, 2007). Waiting increases lead time which is considered a source of competitiveness and reduces end user satisfaction (Womack and Jones 2003).
3. **Transport**: is the unnecessary movement of materials in the factory or on the shop floor (Bicheno, 2000). Womack and Jones (2003) have related this waste with poor communications. Thus, shortening the distance between the customer and the service provider, as well as between the employees themselves, will decrease such waste.
4. **Inappropriate Processing**: is the poor planning of the shop floor that causes the unnecessary movement of work persons and work parts (Bicheno, 2000). For example, having one large machine rather than several distributed smaller machines can discourage operators and create bottlenecks in the long term.
5. **Inventory**: freezing value through the storage of materials or work-in-progress (WIP) or end items. It increases lead times, complexity when it comes to identifying and analysing problems, negatively affects communication, and increases cost.
6. **Motion**: no standards with regard to how to do the work. This leads to unnecessary motion of labour and equipment. Therefore, extra work will be added to each single task (Khalil et al. 2006).
7. **Defects**: this is related to any service or product that does not pass a quality test (Hicks, 2007). This can occur in software development and can lead to

the need for reworking. This aspect can be visualised in terms of scrapping and reworking.

Womack and Jones (1996) have added another type of waste which is the existence of untapped or under-utilised human potential: it refers to a failure to capture ideas or innovations from employees.

By defining the different forms of waste, mapping the value stream of a service project will be easier and more accurate in reducing non-adding value activities. Investigating each activity within the process of a service project will be based on the elimination of such waste.

2.7 How Lean Can Help in Identifying and Improving the Efficiency of Service Operations (Toyota 14 Ways)

Following the implementation of the lean approach in service operations, the research here has adopted the Toyota 14 management principles (Liker, 2004) as a start in terms of guidance to plan, manage, and control a service project, i.e. using a software project as a case study.

1. **Basing the decision on a long-term philosophy instead of aiming for short-run profit**: the production/development process of a system or service is based on a philosophy that encourages independency. Such an approach can be re-used in many similar projects. The design process should be flexible in terms of adding/deleting functional requirements (in the event of alterations in customer's future needs or changes in technology).

Furthermore, consideration to operational issues, e.g. maintainability, customer support, and ease of replacement, are some of the important aspects with regard to long term philosophy as in the software project case under consideration.

2. **Creating continuous process flow to bring problems to the surface**: where it is important to implement a mechanism to deal with, and reduce obstacles and constraints with regard to the movement of information between the different phases (i.e. requirements, design, prototypes, etc.) of the software project. For example, visual management, standardisation, cellular layout of developers, 5S, and easy access to information (i.e. manuals, procedure documentation, and instructions) are some of the mechanisms that reduce constraints within the flow.

   The establishment of the lifecycle philosophy (i.e. creating a repeatable, reliable cycle time for each procedure) helps to expose problems and defects.

3. **Using "Pull" to avoid overproduction**: through adopting a Kanban approach that can be used as a signal to control the flow of information from one process to another. Materials do not proceed from the requirement phase to the design phase until signalled by that station, i.e. the downstream workstation (e.g. testing and integration) controls the flow of materials from an upstream workstation (e.g. coding). This will help to reduce the accumulation of WIP, e.g. un-coded designs, un-tested code, un-documented products, etc.

   The pull technique is based on reversing events, starting from the completion date and moving backwards (Ballard and Howell 2003).

4. **Levelling out the workload**: reducing the batch size (e.g. requirements list, functions in each release), evening out the arrival of work, reducing process tasks, establishing a cadence and adopting standards for the different procedures will all

help to balance the load in different stages of the service system. In addition, prioritising the workload based on customer's needs, can help level out the workload.

Imitating the customer's demand pattern in terms of planning the schedule can help to reduce changeover and can smooth the workload. In a software project, an 'allocation of tasks' on a daily basis, according to demand and vacancies (capacity) of the downstream workstations, can help in levelling out the workload. Furthermore, coordination is an important aspect with regard to levelling out the work, as it is rare that work can be fully partitioned to eliminate interdependencies. Staats and Upton (2009) have suggested the need for architectural simplifications, information clarity, and self-diagnosis to enhance coordination.

5. **Building a culture of stopping to fix problems, to get quality right the first time**: by allowing developers to take actions regarding problems on their sites. This may delay production at first but enhances quality and speed in the long run. Establishing such a procedure increases developers' confidence in taking ownership for fixing defects in their line without any external help.
6. **Standardisation:** this involves measuring the cycle time of different scenarios in accomplishing any new task, and adopting the scenario with the optimal measure. In the case of a software project, adopting a standard form in terms of documenting, naming conventions, writing the code, user interaction conventions, file structure, configuration of management practices, and security standards (Poppendieck and Poppendieck 2007; Staats and Upton 2009). All work should be clearly combined with content, sequence, timing, and output (Staats and Upton 2009). Standardisation reduces the time needed to figure out the best way for executing a task.

Although each new software project has its unique and customised functions, repetitive patterns may exist that can be highlighted through standardising tasks. By identifying similar patterns in each project, and developing common modules (e.g. templates), the learning curve will be sustained (Hines et al. 2006). Hence, standardisation is a key to overcome the complexity of innovatively in product development.

7. **Using of visual control**: having visual control indicators speeds and eases the monitoring of the project's progress. They are early alarms in case the project is biasing from its goals or targets. Different colours to prioritise levels of requirements, risks, etc. can be used.

   Pictures of the standard procedure for accomplishing certain tasks within the service system shorten the lead time and enhance performance. By tiding up the workplace, WIP and unfinished work can easily be identified, and hidden problems (e.g. missing some equipment, or tool failure) become obvious to the workers.

   The adoption of Signalling "kanban" (e.g. using e-mail systems, electronic alarms, etc.) can control the takt time of the process flow in relation to the availability of developers, and the complexity of the task in hand in downstream stations.

8. **Building reliability by using technology that serves employees and processes:** this involves empowering employees by cross-functional involvement and training which will raise their confidence and skills. Therefore, their output reliability will be high.

   Adopting communication methods (emails, video-conference methods, etc.) will help in shortening the distance and increase cooperation between employees.

9. **Growing leaders who thoroughly understand the work, live the philosophy, and teach it to others**: adopting a mechanism for empowering employees through cross-functional involvement will develop a manager who has experienced the workshop environment. Consequently, he will understand its problems, constraints, and solutions more than a manager who has been introduced from a different department. A manager should experience all the tasks that he/she is managing in order to have a clear image of the difficulties his/her employees are facing.
10. **Developing learning employees who follow the company's philosophy:** by sharing responsibility throughout the organisation hierarchy, from top management to the shop floor workers where each level is given the permission to fix their site problems. This means accepting the delay that is caused by such a learning process, when trying different approaches to fix, enhance, or replace a part or produce a product. Accepting change (e.g. introducing new management paradigm, new planning model, structure of the organisation, etc.) by high management supports and guides employees to learn new things and adjust to such change.
11. **Respecting the supply chain and partners**: by focusing on mutual benefits and teamwork philosophy. In helping company's partners understand the company's needs and constraints, they can provide more appropriate answers and solutions to any needs. Adopting short feedback loops with supply chains and partners' increases performance and flexibility in adjusting to dynamic market demands.
12. **No barriers to understanding what is going on the shop floor:** learning from low-level employees who are involved in the work shortens the knowledge loop with regard to what is the true problem on the shop floor, thereby adopting the philosophy that encourages managers to go to the physical location to monitor

operations instead of being told by middle level employees. The connection between the top management and lowest level possible (e.g. developers) should be direct and simple (Staats and Upton 2009).

In addition, stopping the line when a problem occurs allows the company to find the root cause of any problem as close as possible to the source.

13. **Considering all options before making decisions:** testing the different scenarios, possibilities, and risks through the use of simulation tools can increase the probability of a successful decision. Taking the opinions of experts, workers who are familiar with the issue (e.g. operators at a call centre, etc.), and employees who have faced a similar situation, will help to finalise a decision. In addition, having all the necessary data and graphs on one sheet can improve the general vision before arriving at a final decision. Identifying the level of variability of the different operational factors and their effect on success can help to choose a particular decision with confidence.
14. **Becoming a learning organisation through continuous reflection and improvement**: choosing the proper performance measurements, benchmarking, and listening to customers as a means of accelerating improvement. Adopting the proper metrics that suite the operational needs of the service system can reduce time terms of discovering problems, or identifying a lack of balance with regard to resource utilisation. Learning from previous mistakes and building on existing data can accelerate improvement instead of having to start over again from scratch. The company can also use structured problem solving techniques, i.e. scientific experiments that are driven by unambiguous hypotheses for individual change (Staats and Upton 2009).

The previous 14 management principles help in visualising repeated improvements in project operations activities. Moreover, they will be used as guidance to accomplish the following tasks;

a. visualising opportunities of improvement within the planning and control activities and identifying weakness within existing planning models in literatures,
b. choosing the best practices from the existing software development models that align with the principles recommendations and suggestions,
c. visualising and identifying the variability elements or factors within the software project activities during its lifecycle,
d. forming the research methodology to be taken within this research, and
e. designing the standard steps that are proposed by this research to plan and manage the service project.

Chapter 3: Service Operations

3.1 Introduction

This chapter illustrates the generic characteristics of service environment that governs its operations. Moreover, the research has used software development project as a case study. Therefore, the existence software development models were investigated to help develop the proposed standard model and choose best practices out of them. Lean principles in software, Quality Function Deployment (QFD), operations factors, voice of customer, and risk management are all illustrated as they build the pieces of the enhanced planning model the research here has developed and will be illustrated in the coming chapters.

3.2 Characteristics of Service Operations

There exist some unique characteristics that differentiate service operations from other sectors (Looy et al. 2003; Meredith 1992; Kosuge et al. 2009). These characteristics are presented in Table 3.1. They help highlight the specific constraints that may affect planning activities;

a. **High customer contact** and active consumer participation in the process: the customers present to some limit improve service system performance and consume the service system resources (Arbos 2002; Spohrer et al. 2007).

b. **Labour intensive**: personnel strategies are critical in services project's success. In addition, personnel consume high percentage of the capital (Wikstrom et al. 2009). Reducing size of employee by cross functions and automation will reduce cost.

c. **Intangible, variable and nonstandard output**: the output of a service system is usually in the form of actions that create values toward final customer. These actions should be standardised based on best practice in satisfying customer (Arbos 2002; Greasley 2006; Brentani 1991).

d. **Difficult to measure service quality and productivity**: because the goods

(products) are intangible and abstracts. Different standards for services quality can be used. This increases the difficulty of finding best service from customer perspective (Karmarkar and Pitbladdo 1995).

e. **Psychological benefits**: the satisfaction of the customer after executing the services operations is a major factor of success. It is proportional with quality, efficiency, and effectiveness of the services provided (Arbos, 2002).

f. **Dependency on skilled personnel**: the highly skilled personnel in an organisation, the better the output and performance achieved. Therefore, their number and field of expertly play major role in cost, success, and risk if they are absent.

Table 3.1: Characteristics of Services System

3.3 Software Project

The research aimed at developing a model to improve the planning and control of high dynamic service project. As mentioned earlier, the research chose software project to be a case study as software is a product development which involves diversions tasks (Petter, 2008).

3.3.1 Software Process

Kouskouras and Georgiou (2007) defined software development process to be "*a set of activities, methods, practices and transformations used to develop and maintain the software and relevant products (e.g. project plans, design documents, codes test cases)*". Processes exist in all form of manufacturing and services as hidden structures of correlated activities. They capture the link between customer needs and customer satisfaction as presented in Section 1.3 (Ljungberg, 2002).

Software project is of customer-centric type where the focus is on delivering customer solutions consisting of both goods and services (Wikstrom et al. 2009; Chen and Wang 2009; Hoch et al. 2000; Silver 2004; Amescua et al. 2004).

3.3.2 Software Project Process Mapping

There exist different approaches that convert the 'raw' requirements into a software system. Some approaches are strict and use disciplinary rules in developing business solutions or products to the customer; whereas others are more flexible and iterations are allowed with risk management as will be shown in Section 3.4. In the latter approach, the development of the product takes an incremental behaviour where its parts are being delivered piece-by-piece.

To identify the different levels of variability that may occur in the operational process of software project, it is important to map down all the activities within. The research here has used the following methods to map down the software project process:

a. reviewing the literature,
b. analysing activities occurring in the work breakdown structures as in Table 3.2,
c. formal/informal interviews with project managers and team leaders of ADNOC Company as will be explained in more details in chapter 4, and
d. develop process mapping to breakdown the tasks involved in every phase of software project development.

Accordingly, seven stages are being identified that represent the lifecycle of the software project and are presented in Table 3.2. Moreover, the different activities of

software project are being presented based on previous methods, e.g. Boehm (1981), and ISO 12270.

Software Process Phase	Activities
Requirement	1. Validated system architecture hardware-software, operation tasks include human-machine allocation. 2. Validated software requirements specification (functional, none-functional, performance, interface interaction). 3. Validated development plan, milestone delivery, resource management, team member responsibilities, and schedule. 4. Validated product development control plan, configuration management plan, and quality assurance.
Discussion	1. Feasibility study. 2. Resource allocation. 3. Budget allocation. 4. Control Change Responsibilities and Procedure.
Design	1. Detailed design specification: a. Program component hierarchy. b. Data base structure. c. Validated completeness, consistency, feasibility and traceability to requirements. 2. High risk analysis. 3. Preliminary integration and test plan.
Prototype	1. Validated detailed design specification for each unit: a. For each routine (≤ 100 LOC) specifies name, purpose, sizing, error exits, inputs, outputs, and processing flow. b. Data base description (attributes and types of characters). 2. Descriptive project without any functionality.
Code	1. Completion of unit-level as documented in requirement. 2. Executable module level systems.
Test & Integration	1. Verification of all unit computations, input and output options, error messages, etc. 2. Verification of programming standards compliance. 3. Verification of acceptance test. 4. Integrating sub-modules to complete system.
Deployment	1. Satisfaction of system acceptance test. 2. Completion of all specified conversion and installation activities. 3. Calculation of cost and comparison to the estimated plan.

Table 3.2: Software Development Lifecycle Activities (Alkaabi et al. 2009)

The previous mapped down stages will be used to build the simulation model as will be shown in Section 4.4 Step 3. The simulation model is part of the proposed model and adds automation feature to it.

3.4 Software Development Models: Strengths and Weaknesses

Software Development model is the framework that is concerned with defining, structuring the process of developing a computerising/informatics system. Whereas software project is a discipline to plan, control, monitor the project. The software project may adopt or follow one or many software development models to achieve its goal based on the various technical, organisational, project and skills considerations. Software Development dates back to the mid-1960s (Boggs, 2004). Since then different models had been developed. The variations of these models is linked to the variations in project's goals (e.g. cost, quality, customer satisfaction levels, size of team, etc.), resources availability (team size, skills' level), application domains, and risk's degree. This section covers some of the major software development processes and states some of their strengths and weaknesses in relation with Toyota 14 ways in Section 2.7.

3.4.1 Pure Waterfall Model

Waterfall is the first formal software development process that still in use by many organisations to this day (McConnell, 1996). The Waterfall model is a sequential software development model developed by Royce (1970) where iteration is prevented. The steps of the Pure Waterfall Model are presented in Figure 3.1.

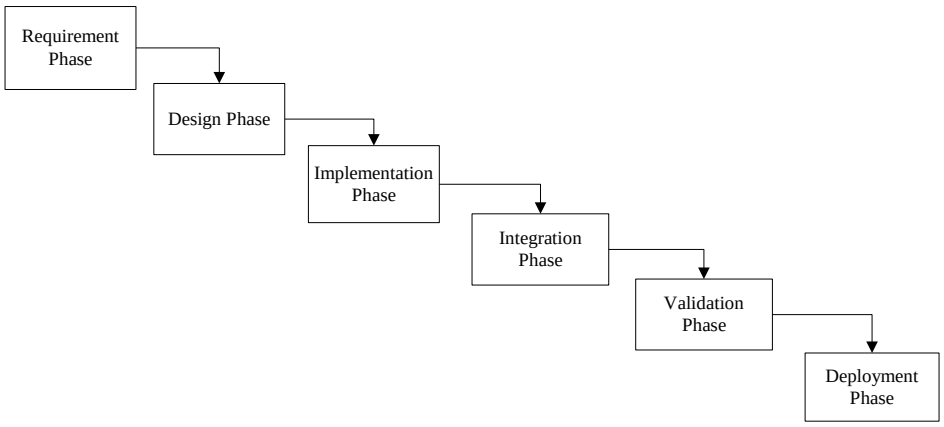

Figure 3.1: Stages of Pure Waterfall Module

Analysis to this model discloses some of its strength and weaknesses as shown in Table 3.3.

Strengths	Weaknesses
a. strong documentation practice (McConnell 1996; Vliet 1993), b. proper model for stable projects (Highsmith and Cockburn 2001), c. proper for critical projects where backup files is a must (McConnell, 1996), and d. well defined and simple sequential linear process.	a. inflexible for not allowing iteration (Morien 2005; Collyer and Warren 2009), slow pace process due to requirements setup (Molokken-Ostvold and Jorgensn 2005), b. no output in the first stages except documents, and effort consuming in updating the written document, c. no utilization of teams' members in most of the project's time i.e. if the specification team is busy, then the design team will stay idle, and d. no risk assessment.

Table 3.3: Pure Waterfall Model Strengths and Weaknesses

3.4.2 Spiral Model

Spiral Model combines prototyping and designing in its stages, where iteration is allowed in contrast with waterfall model principle (Boehm, 1988). Although spiral model was not the first to mention iteration, it was the first model to emphasize on the importance of iteration in decreasing cost and errors (Vliet, 1993). It is proper for large, expensive and complicated projects. Spiral Model is risk-driven rather than document driven (Boehm, 1988).

Risk analysis in each iteration and presentation of a prototype to customer is a way of discovering major risks in earlier stages of the project (McConnell, 1996). Thus, this is a good practice to eliminate rework, waste and bugs from escaping to the enhanced model. Figure 3.2 illustrates the stages each iteration pass through.

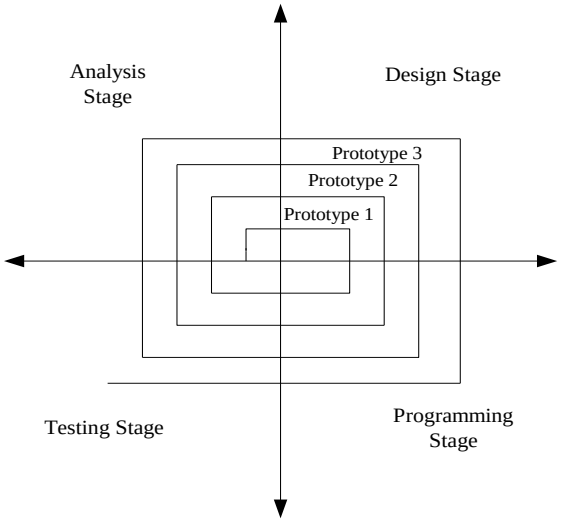

Figure 3.2: Spiral Model Stages (Boehm, 1988)

There are some advantages and disadvantages in spiral model as presented Table 3.4 below.

Strengths	**Weaknesses**
a. first to introduce prototyping and risk analysis (Boehm, 1988), b. flexible in termination after risk analysis step at each iteration, and c. time and cost estimations are more realistic (Andersson and Runeson 2007).	a. costs and time consuming (McConnell, 1996), and b. complex process due to the many steps in each iteration.

Table 3.4: Spiral Model Strengths and Weaknesses

3.4.3 V-Model

This model has a strict validation practice. Inspections are executed to each sub-unit as an entity that helps eliminating bugs. The testing points are (Hughes and Cotterell 2006);

a) unit,

b) system, and

c) acceptance.

The 'V-model' is developed in Germany in summer 1992 (Plogret, 1996). Its name is related to the shape of its phases as shown in Figure 3.3.

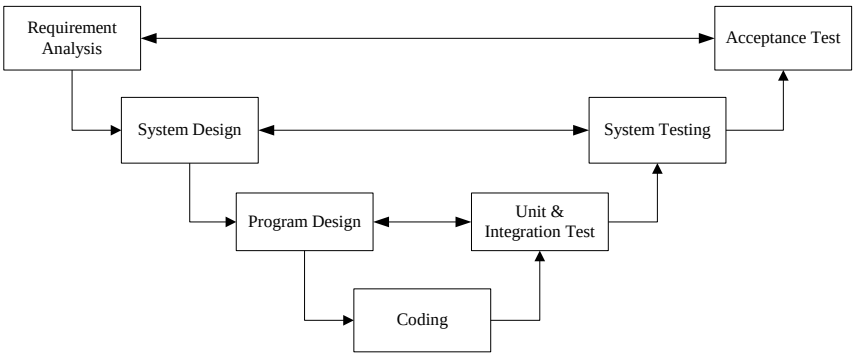

Figure 3.3: Stages of V-Model (Plogert, 1996)

The following table presents the identified feature about the V-model:

Strengths	**Weaknesses**
a. concrete assistance in the project lifecycle, and b. concrete validation scheme that helps in project inspections earlier to implementation (Pyhajarvi et al. 2002).	a. slow and time consuming process, b. addresses software development within a project rather than a whole organization, and c. document-driven model similar to waterfall, which consumes time.

Table 3.5: V-Model Strengths and Weaknesses

3.4.4 Code-and-Fix

The Code and Fix model has no formal structure or pre-stage preparation. The developer starts coding the product without any analysis to time, cost, documentation and plan (McConnell 1996; Yilmaz and Chatterjee 1997). It consists of two steps:

1. Writing some code.
2. Fixing the written code.

Code-and-Fix Model Strength: the only strength in this model is its ease of use to single developer teams (Boehm 1988; McConnell 1996).

Code-and-Fix Model Weaknesses: despite its easy and simple use, there are some weaknesses in it such as:

a. the structure and efficiency of code is lost after many iterations of fixing,
b. high cost and effort in fixing the codes (Boehm, 1988), and
c. not applicable for middle and large projects as any late change means restarting from scratch, and not a team-base model.

3.4.5 Evolutionary Prototyping

This model starts with the development of a basic concept set of the product. Then it incrementally adds to the basic set more visible and tangible operational features based on user requests. After the addition of a new batch to the prototype, it is being presented to the customer to get his/her feedback and assess the needs (Greer and Ruhe 2004; McConnell 1996).

This model helps a type of customers that are not sure what exactly they need in their system (Boehm, 1988). Evolutionary Prototyping is useful with projects that have rapid requirement changing and for a fast delivery needs (Stephen and Bates 1993). In this

iterative version model, only the design (prototype) cycle is repeated to gather knowledge and check customer satisfaction percentage, while the main execution phase is carried out only once (Collyer and Warren 2009).

The strengths and weaknesses of this model are presented in Table 3.6 below:

Strengths	**Weaknesses**
a. proper for rapid requirements changing (Stephen and Bates 1993), b. short feedback loop through fast delivery of output results to customer (Pretschner et al. 2003), c. customers can start benefiting earlier from the system and provide their feedback (Greer and Ruhe 2004), and d. important requirements are delivered first while less important requirements are scheduled to be delivered later if the budget and schedule is sufficient enough.	a. lack of control, as freedom is granted to users to continue change specifications in each release (Stephen and Bates 1993), b. the final product contains unstructured and inefficient code 'Spaghetti code' because of lack of planning and rapid changing (Boehm 1988; McConnell 1996), c. uncertainty is high in schedule due to the unknown ending time of the prototyping phase (McConnell, 1997), and d. conflict between stakeholder and technical requirements' (Greer and Ruhe 2004).

Table 3.6: Evolutionary Prototyping Strengths and Weaknesses

3.4.6 Staged Delivery or Incremental Implementation

Stages Delivery is based on the same concept of previous model presented in Section 3.4.5. However, the number of stages and functionality to be delivered during each stage is well planned and agreed on by both developers and customers (McConnell, 1996). It is a top-down approach where minimal functionality is developed at first and more requirements/functionalities (i.e. more detailed ones) are added in each successive increment (Greer and Ruhe 2004).

Table 3.7 presents the strengths and weaknesses of this model.

Strengths	Weaknesses
Help the end user to benefit of the system through the fast delivery of most fundamental functions as requested in earlier stages (Barney et al. 2008).	a. A need for a well-defined delivery plan that considers the functionality dependencies (McConnell, 1996), and b. A standard method to deal with frequent requirements changes (Boehm, 1988).

Table 3.7: Incremental Implementation Strengths and Weaknesses

3.4.7 Design-to-Schedule

This software development model is similar to the previous model 'Staged Delivery Model' in the concept of breaking the project into pieces and delivering them in a successive way. However, the big difference is the restriction in the project schedule i.e. if the plan is to deliver the project in 9 stages and gets delayed in one stage more than planned, the project will be delivered with 8 stages or with the only finished stages. Therefore, there is a need for clear plan and control of each stage which the research here has considered while developing a proposed standard model, (McConnell, 1996).

3.4.8 Comparison between Software Development Models

Although the presented software development models share the same necessity and timing constraints; each model has its own unique features that meant to deal with certain projects in the time frame of the projects (McConnell, 1996). In summary, Table 3.8 below has been adopted from McConnell (1996) and the research has extended it to make a full comparison between the different software development processes according to specific factors related to their use.

By identifying the existence models in the software development field, the research looked at the strength and weakness points of each model and briefly adopts best practices of each model. In addition, it will also adopt the lean philosophy in highlighting the value and non-value activities among software project phases and risk factors associated with each. Lean is the philosophy for a long term improvement (Gunasekaran and Yusuf 2002) whereas agile is driven by being highly responsive while compromises cost-efficiency. This agrees with Toyota 14 ways (i.e. 1st way) mentioned in Section 2.7 of having long term improvement.

The different existing software models that have been discussed earlier along with lean software development model will help in developing the proposed model. They provided the following assistances:

a. More understanding and clarifying of the software environment to identify constraints (factors) that affect project outcome as will be shown in Section 3.9.
b. Mapping down the software project as will be shown in Section 4.4.
c. Adopting best practices from the existence software models as will be shown in Section 6.2.

Factors	Waterfall Model	Spiral Model	Code-and-Fix Model	V-Model	Evolutionary Prototyping	Staged Delivery Model	Design-to-Schedule	Lean Software Development
Ambiguous Requirements	Poor	Excellent	Poor	Fair to Excellent	Excellent	Poor	Poor to Fair	Excellent
Unclear Architecture	Poor	Excellent	Poor	Fair to Excellent	Poor to Fair	Poor	Poor	Excellent
Produces Reliable System	Excellent	Excellent	Poor	Excellent	Fair	Excellent	Fair	Excellent
Allow Change to Size and Scope	Excellent	Excellent	Poor to Fair	Poor to Fair	Excellent	Excellent	Fair to Excellent	Excellent
Manage Risks	Poor	Excellent	Poor	Fair to Excellent	Fair	Fair	Fair to Excellent	Fair
Can be Constrained to a Predefined Schedule	Fair	Fair	Poor	Fair	Poor	Fair	Fair	Fair
Has Low Overhead	Poor	Fair	Excellent	Fair	Fair	Fair	Poor to Fair	Excellent
Accept Middle Stage Corrections	Poor	Fair	Fair to Excellent	Poor to Fair	Excellent	Poor	Poor to Fair	Excellent
Progress Visibility	Poor	Excellent	Fair	Excellent	Excellent	Fair	Fair	Fair to Excellent
Provide Management with Progress Visibility	Fair	Excellent	Poor	Excellent	Fair	Excellent	Excellent	Excellent
Is Training Needed to Use the Model?	Fair	Poor	Excellent	Fair	Poor	Fair	Poor	Fair

Table 3.8: Software Development Processes' Comparison (McConnell 1996; Alkaabi et al. 2009)

3.5 Lean Principles in Software

Lean principles as presented in Section 2.6 are adapted to software sector with attention to the specific characteristics in Table 3.1 that govern any software project. The research here analysed Lean Software Development Model (LSD) and used some of its features in developing the proposed plan and control method.

Through Literature review, works with centre of attention pertaining to software development and lean thinking did not appear until late in the 20th century (Raman 1998; Perera and Fernando 2007). Most of this literature is produced by Poppendieck (2002) (Sutherland et al. 2008; Parnell-Klabo 2006; Morien 2005; Tonini et al. 2007).

3.5.1 Lean Software Development (LSD)

The transformation of lean principles in Section 2.6.1 into software projects creates what is known to be Lean Software Development (LSD) as shown in Figure 3.4. The LSD consists of principles that comprehend the benefits and value to be attained, in addition, to tools and practices to achieve these benefits and values.

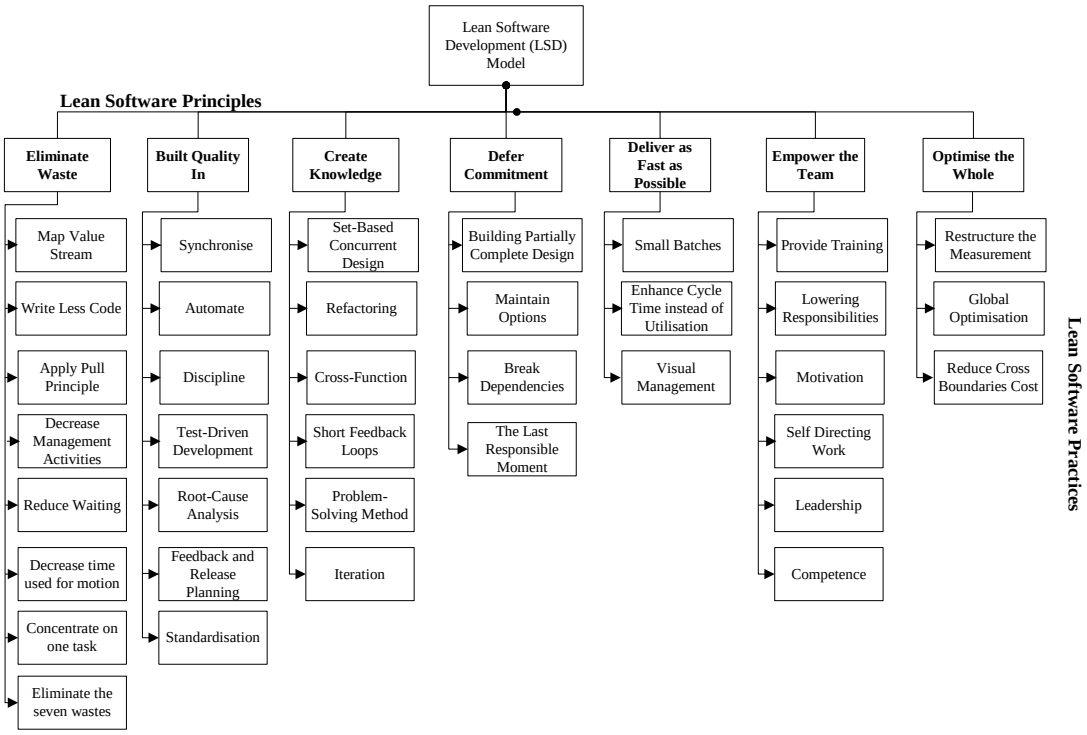

Figure 3.4: Lean Software Development Model

The principles of LSD are:

Principle 1 Eliminate Waste

This principle is the core for successful Lean implementation (Hicks, 2007). The first step in eliminating waste is to identify value (Kumar, 2005). LSD eliminates waste, along with the practices in Figure 3.4, by:

a. **Mapping the value stream:** it is important to visualise the whole activities within the development process; then divides the activities into three groups, added value, non-added value but core, and non-added values neither core activities (wastes) (Alkaabi et al. 2009).

b. **Applying 'Pull-Principle':** the designers pull information from the analysts, coders pull information from the analysts, etc. (Boot et al. 2008). Each phase in downstream is pacing according to the needs of the upstream phases. This is also recommended by the Toyota 14 ways in Section 2.7.

Principle 2 Build Quality In

This principle encourages having quality embedded in the process than in the product itself. LSD implements this principle with the tools shown in Figure 3.4, the research here has focused on the following to be adopted in the proposed model;

a. **Automate** any step in the process that produces quality and make it a daily routine, e.g. testing, installation, etc. (Gunasekaran and Yusuf 2002). Using simulation can automate the action of validating the different combinations of variables affecting the planning activity as will be presented in Section 4.4. It supports the Toyota 14 ways in Section 2.7 of stopping and fixing the problem automatically.
b. **Root-Cause Analysis**: is asking 'why' five times. This tool is been applied in software projects to enhance the quality of final product (Nelson et al. 2000).
c. **Standardisation:** it helps ensuring consistency in accomplishing the required task in same manner (Liu et al. 2008; Allway and Corbett 2002), eases the identification of cause and effect (Staats and Upton 2009; Spear 2006), and accelerates work flow (Allway and Corbett 2002). This aligns with Toyota 14 ways presented in Section 2.7, and with the research aim in developing a standard practice in planning projects as will be shown in chapter 6.

Principle 3 Create Knowledge

Systematic learning can preserve knowledge and helps making decision based on the retained knowledge (Poppendieck and Poppendieck 2007). This can be done through

a. **Simultaneous and Set-Based Concurrent Design**: it is a method to enhance learning by providing multiple set of solutions to the problem in concern. Then

eliminating the non-ideal options (solution) as time passes and uncertainty decreased, until a clear optimal choice can be made (Sobek II et al. 1999). Accordingly, different plan scenarios can be provided and eliminate the non-ideal ones during execution of project. In this manner the different choices are being examined before making the decision as supported by Toyota 14 ways in Section 2.7.

b. **Cross-Functional and Cross-Team Culture:** increases skills, motivation and communication by working on different tasks, experiencing new working environments (Furugaki et al. 2007; Kindler et al. 2007; Martine 2001). Knowledge is shared as a result, no critical member is causing bottleneck in the flow. This is also been recommended by Toyota 14 ways in Section 2.7 and assumed during the development of the simulation model.

Principle 4 Defer Commitment

The aim is to delay locking the design or architecture or any part of the system to the last moment which if it is delayed more; an external factor will clear the uncertainty (Poppendieck, 2002). Commitment to irreversible decision is deferred to be based on a known event than on forecasts (Ballard and Howell 2003).

LSD provides the following practices to defer commitment (Poppendieck and Poppendieck 2007):

a. **Building partially complete design**, delivering small batches frequently, and having most decision reversible.
b. **Maintain options:** through providing set of options and choices for unalterable

decisions based on their impact upon the project.

c. **Break dependencies**: having independent features help adding them when they are needed and with any order.

Principle 5 Deliver Fast

The shorter the interval time between deliveries, the far ahead decision can be made. This helps the process to adjust to customer changes faster. Assigning more time to uncertain and potential variable tasks while accelerating the phase completion date is another approach that helps in delivering fast (Ballard and Howell 2003). LSD stipulates smooth, efficient, rapid flow in response to customer request through:

a. **Small batches**: reducing the size of delivered batch as small as possible helps flow and reduces cycle time (Kilpatrick 2003; Kindler et al. 2007).

b. **Enhance cycle time instead of utilisation**: high capacity utilisation builds up queues, inventory and increase cycle time of process. Reducing cycle time is the key to eliminate such queues especially in process with variability as software development (Reinertsen, 2005).

Principle 6 Empowering Team Members

Empowering team comes through training, expertise, motivation, administration support rules, leadership and champion (Poppendieck, 2002, Kindler et al. 2007). Leadership is accepting the responsibility to proceed to success, in which teamwork is essential but requires to be synchronised in right direction with responsible leadership. This agrees with the characteristics of the intended plan and control model presented in Table 2.1 as well as suggested by Toyota 14 ways (i.e. principle 9 and 10) of growing leaders from

the team members, and empowering the learning process of those team individuals. Moreover, this principle supports the concept of process orientation improvement mentioned in Section 1.3.

Principle 7 Optimise the Whole

Focusing on part of the mapped process will cause the rest to be de-enhanced (Parnell-Klabo 2006; Shinkle 2005). Breaking the value stream into pieces and optimises them individually then recombine them back will not cause optimisation of the value stream (Poppendieck and Poppendieck 2007). To optimise the whole:

a. **Restructure the Measurement**: allocate the measurements that reflect the performance of the whole value stream, e.g. %waiting, %rework, instead of parts of the value stream as will be shown in Section 4.4.
b. **Reduce Cross Boundaries Cost:** having the culture of one management system increases speed of flow. Most of big delays occur at department boundaries (Poppendieck and Poppendieck 2007).

Some of the previous practices and tools are adopted in the research methodology and in developing the proposed method aimed at this research as will be shown in coming chapters.

3.5.2 Software Development Seven Wastes

Through literature reviews, the manufacturing seven wastes are in different forms with software projects as presented below in Table 3.9 (Poppendieck and Poppendieck 2007).

Manufacturing Waste	Software Development Waste
In-Process Inventory	**Partially Done Work** such as un-coded documentation, un-synchronised code, un-tested code, un-documented code, un-deployed code, design-in-process documents, and etc. They are the inventory of software development (Poppendieck and Poppendieck 2007). The difficulty in such inventory is its invisibility; they can be saved as electronic copy in a hard disk (McManus and Wood-Harper 2007; Tonini et al. 2007; Reinertsen 2005)
Over-Production	**Extra Features** is producing extra functions that are not required by the customer, e.g. unnecessary code and functionality. Statistics show that only 20% of the delivered functions are being used by the end users (Whittaker 1999; Smith and Reinertsen 1992).
Extra Processing	**Relearning** is the loss of knowledge that has been learned through experience and problem solving (Poppendieck and Poppendieck 2007). Regenerating the same knowledge over and over is a major waste of time, resources and motivation (Liker, 2004).
Transportation	**Handoffs,** software development knowledge is a type of tacit knowledge that is hard to be transferred through documentation alone to another developer without losing major partial of the knowledge.
Motion	**Un-Focusing** results in distracting work flow; wastes time to re-adjust to the new task and often detract results of both tasks. Interruptions such as unnecessary meetings, announcements, sickness, breakdown, etc. are the ultimate enemy of deep concentrated thinking. Having cross-function teams reduces such waste by; **a.** having the team share same level of skills, and **b.** no critical developer phenomena, i.e. his/her work needs to be stopped and switch him to fill in another task as no one in the team can do so.
Waiting	**Delays** occur when waiting for resources (developers, artefacts, etc.) to be available, and decision of customers and management (Kumar, 2005). Slow internal communications can cause waiting. Unclear requirements, less customer involvement especially in earlier stages, less top management support or bureaucracy can increase the %waiting.
Defects	**Defects** are bugs escaped the testing phases. Bugs reduce quality of final products, cause reworking, and stop the line to be fixed (Poppendieck, 2002). Early tests and adopting test case scenarios can reduce defects (Morien, 2005). Insufficient testing will lead to stoppable process repetition.

Table 3.9: 7 Wastes in Software Development (Poppendieck and Poppendieck 2007)

3.6 Project Management Operations

Project management is a discipline of planning, controlling, organising, and managing the different types of resources (e.g. personnel, equipment, materials, etc.) to achieve the specified project goals (Laslo 2010; Vidal and Marle 2008; Anderson and Merna 2003). Laszlo (1999) described project management to be a universal management tool that can be fitted to any type of projects. It ensures all stages of the project follow a

systematic approach of careful planning, monitoring, and measuring (Murphy and Ledwith 2007).

Any project has its own internal structure that consists of resources, deliverables, tools, milestones, etc. Through time progress, resources are consumed, members are changed, experience is gained, and products are delivered without project losing its own identity. In real projects, the starting point might have had a definite starting point; nevertheless, the scope keeps changing and leads the ultimate completion date to slide out further. Thus a clear plan to take in advance such complication is important (Ebert 2007; Atkinson 1999; Laszlo 1999; Horman and Kenley 1996).

There is still a controversial issue in relation with the influence of tools, methods, instruments, individual qualities, and of project members' skills among the different phases of a project. In that sense, Rose et al. (2007) pointed out that success is placed on shoulders of the people involved, project members, leaders and flow of information. Their results were based on qualitative analysis (interviews, focus groups, and plenary meetings) on medium size software projects.

Moreover, high percentages of project managers do not apply any technique in managing their projects; and few apply only 'Project Management Software' and 'Gantt Charts' (McBride 2008; White and Fortune 2002). This lack of use can be root caused to the complexity and time consuming exits in recent models where this research is aiming to avoid.

3.6.1 Existing Methods of Software Project Management

Different researchers investigated the software industry business operations. Improving outcome of software project operations took different forms, e.g. problem-oriented approaches, appreciative inquiry approach (Holmberg et al. 2008), product line/factory approach (Ahmed and Capretz 2007), reducing cognitive bias while system designing through the use of traceability and anchors forming (Mohan and Jain 2008), reducing inconsistency of the system's modules with a set rules (i.e. ignore, defer, circumvent, and ameliorate) (Nuseibeh et al. 2001),

To overcome the limitation in applying earned value analysis (EVA) in managing software operations; Li et al. (2008) developed an evolutionary Work Breakdown Structure using case points approach. Other researchers went to identify the best software project management practices (Maqsood and Javed 2007, Verner and Evanco 2005).

Wagner and Durr (2006) have developed a five-step method for value-based planning and monitoring of systems engineering projects that is similar to QFD. However, no priority technique was used in this method and no consideration to risk was given too. The priority in the research here will be in the form of how to react to the most influences constraint that may affect projects deliverables and completion date.

Theory-W is a classical software project management framework that had considered risk analysis, different stakeholder, customer requirements, and maintainability of the product (Boehm and Ross 1988). It had rolled the project manager as a negotiator

between the different stakeholders needs'. Nonetheless, it is effort consuming to apply this framework.

Still the previous approaches tend to lack in investigating the success factors that impact on project performance over its lifecycle (Pollack 2007; Dvir and Lechler 2004). The research here is aimed at improving the plan and management of the project under the use of lean philosophy. Therefore, attention is paid to adopt the structure and concept of the Quality Function Deployment (QFD) as it is one of lean tools dealing with project management, planning, and control where a focus on customer requirements is emphasised.

3.7 Quality Function Deployment

QFD is a method which had its origin in Japan during the 1960s by Akao and Mazur (2003). It is a comprehensive quality tool that focuses in capturing customers' requirements (Akao and Mazur 2003; Karlsson 1997). QFD helps in bringing multifunction teams of different sections to cooperate for finishing customers' needs (Bouchereau and Rowlands 2000; Gonzalez et al. 2008; Chan and Wu 2002).

Through literature review, QFD has succeeded in capturing the voice of customer and accelerating the development and service operations toward achieving such objective (Chan and Wu 2002; Prasad 1998). Therefore, QFD can be used as a pro-active customer-driven planning and control tool to solve problems earlier in project operations before they escaped to downstream activities.

According to literatures, many methodological works have been integrated with the original QFD to efficiency its output, extend its use, and simplify its procedure e.g. quantitative methods are being used to increase QFD's reliability and objectiveness (Hari et al. 2007; Bouchereau and Rowlands 2000).

Different researchers applied QFD to improve the output of their processes (Hari et al. 2007). Stuart and Tax (2006) had shown that QFD can be used as an effective service planning tool in both strategic and tactical levels as trade-off decision made based on service encounter interrelationships. Their approach consists of three metrics, service bundle planning based on customer requirements, service delivery development that specify the process elements, and finally service process control to sustain high quality of service based on the indicators values.

3.7.1 QFD in Software Operations

In relation with software needs, Software Quality Function Deployment (SQFD) models start with functional/behavioural requirements rather than quality one which can affect software project success. For example, Zultner (1993) has developed a Software QFD model that is divided into three parts; each part consists of some metrics. The three parts are:

a. Fundamental Deployments that consists of two metrics, Customer Deployment and Quality Deployment (Zultner, 1993),
b. Horizontal Deployments that consists of three metrics which are function deployment, information deployment and task deployment (Liu et al. 2006), and

c. Vertical Deployment that consists of three metrics, which are technology deployment, cost/schedule deployment, and reliability deployment (Zultner, 1993).

Another Software model is Shindo's model, which is composed of three main tasks, functional definition, sub-system models, and developing a complete specification for each sub-system i.e. interface, functionality, data base, etc. (Herzwurm and Schockert 2003).

Herzwurm and Schockert (2003) combined the advantages of previous models through PriFo Model. 'PriFo' stands for 'Prioritising and focusing', which is the major aim of the model. The model consists of five steps that include (Herzwurm et al. 2003);

a. meeting all the different stakeholder groups to identify time schedule, cost, and goals,
b. prioritizing and weighting the requirements using affinity and tree-diagrams techniques,
c. customer's requirements are gathered and are being input to a table represent voice of the customer,
d. technical voice of the engineer is input to another table, and
e. both tables are combined together to create the classic House-of-Quality (HOQ) matrix. The model processed after this step in the same classic QFD steps.

The previous models lack in automating the procedure for an ease to use in large size projects. For example, although Zultner SQFD is comprehensive in gathering all relations between high-level customers' need and low-level technical characteristics; the

model is complex and takes a great amount of time to be established, especially with middle to large size software projects. Success in project delivery does not depend solely on meeting customer demands; however, competition place, current performance and customers' perspective are important factors to be included in prioritising, planning and control of any project (Lai et al. 2008).

A survey to study the effect of adopting QFD in major software vendor (MSV) has showed a significant positive impact on quality and productivity (Haag et al. 1996). A group decision making approach is being implemented by Buyukozkan and Feyzioglu (2005) that makes use of multiple expression formats of QFD to capture and analyse the different demands of customers. In their approach, two techniques have been used to overcome two major obstacles in other SQFD models; group decision making techniques is been used to prioritize customers' requirements (CRs) into design requirements (DRs), and fuzzy set theory is been used to clear the inherently vague human linguistic expressions of customers' demand.

Moreover, SQFD had been integrated with Goal-Question-Metrics (GQM) to enhance the quality of software development (Pai, 2002). This methodology satisfies the needs of both customers' requirements (e.g. end customers, managers, and developers) and project goals. Pai's methodology consists of five steps:

a. Recording customer requirement in their words.
b. Identifying project goals that had been collected from project stakeholders.
c. Raising questions based on project goals and their answers align with customer requirements.

d. Re-organise the technical requirements (HOWs).
e. Building the technical priority based on the summation of the multiplication of customer requirements priorities.

This methodology agrees with the aim of this research in prioritising the technical requirements based on customers' needs and project goals. Nevertheless, there is no clear consideration to risk factors that the project can experience throughout the development lifecycle.

Lack of automation and risk consideration are some of the weaknesses in previous models that will be overcome in this research.

3.8 Customer Demands

The current research is taking customer demands as an input and a starting point to help plan and manage any size project. They are the first part of the proposed model as will be shown in Section 4.4. A clear understanding of customer demands and the capability of the resources in hand is important in setting and implementing the plan and control of the project.

Moreover the first phase of any new product/service development lifecycle is getting the proper requirements. According to literature review, the requirement process/phase goes through the following steps; elicitation, analysis, specification, and validation (Dorfman, 1997). In addition, it is important to understand that the service/product requirement is not a discrete front-end activity; however, it is a continuous process that initiates at the beginning of the project lifecycle and proceed with refinement to the later

stages. Therefore, the proposed plan is recommended to be applied through different phases of the project lifecycle in a continuous fashion.

3.8.1 Voice of Customers

It is the exact customers' needs in their own words, which is the first principle in lean thinking and insures his/her satisfaction. Customer demands/requirements are essential factor in the success of any service process. Many researchers define quality of a service product to be the degree of having the exact customer's requirements (Lauesen 2003; Ahmed and Capretz 2007; Knod and Schonberger 2001).

Moreover, customer demands can experience degree of volatility or change due to requirements change (i.e. new technology are introduced, more understanding of the problem, new needs exist, etc.), thus they need to be under continuous monitoring through the lifecycle of the project (Ebert, 2007). Understanding their needs earlier in the service project lifecycle helps the planning and control process and reduces rework or effort wasting (Charnes and Cooper 1961; Zapfel 1998; Williams 1985; Tyagi and Das 1999).

3.8.2 Customer Demands Prioritisation Techniques

Different techniques are used to solve conflict between stakeholders demands such as requirements prioritisation and business case analysis techniques, and stakeholder identification and requirements negotiation techniques (Boehm, 2005). To achieve maximum value of satisfaction to customers, the value-based approach suggests the

alignment of product, project and business decisions, and involvement of the various stakeholders; while prioritising and selecting their demands (Barney et al. 2008).

Prioritising techniques through literature are varied from high-level to detailed prioritisation algorithms. The Planning Game is an extended way of doing numeral assignments with ranking. However, it is time consuming (Berander et al. 2006). Hierarchal-Analytical Hierarchy Process (AHP) is similar in structure to classic AHP with the difference that the most generalised requirements are positioned on the top, whereas the more specific requirements are placed at lower levels of the hierarchy. It is not as flat structured when compared to AHP and it is more sensitive to judgement errors (Karlsson et al. 1998).

The proposed method will capture customer demands and use them as the targets or goals the project is supposed to achieve. In that sense, the plan will be built or suggested by the proposed method to satisfy such demands. Therefore, optimum solutions will be provided in the case that outcome of present values of project environment are not matching customer demands as will be shown in Section 4.4.

3.9 Identifying the Operational Factors (variability) of the Software Project

One of this research aims is to identify the different types of variability within the service operations. Variability cannot be eliminated but it can be reduced to minimum levels so that the final performance is high.

Through literatures, different factors or variables had been identified that raise the uncertainty level in planning and managing the software projects (Heiat 2002; Fenton and Neil 1999; Graves et al. 2000; Huang and Lo 2006). Wu (2006) identified design tool, developer skills, expertise, team size, program complexity that are considered as major variables to affect the process. Other Researchers identified some variables that can be considered as cost drivers, i.e. the COnstructive COst MOdel (COCOMO II) (Baik et al. 2002; Boehm 2000; Boehm 1991). These variables are used to estimate effort, schedule, and cost of the project under consideration.

Through this research and based on the improvement steps suggested by Toyota 14 ways in Section 2.7, generic operational factors were identified as follows:

1. **Size of Team:** is the number of developers, analyst, designers, etc. involved in the development process of the system. Balancing this factor is important as the larger the team size, the more communication overhead exists (Lin et al. 1997). This factor plays a major role in cumulating the overall cost of the project (Hazzan and Hadar 2008). The research grouped the different types of developers in a pool group that support the different activities throughout project lifecycle. This action agrees with lean principles in having cross function teams and to simplify experimentation as in Section 3.5.1 Principle 2.

2. **Skills and Experience:** refers to the knowledge about the system domain, tools, programming languages, standards, and logic that is needed to finish the system. The more experienced the team members, the rapid the development process, the

less training and support needed. This factor includes the learning factor that is considered by many researchers as the time being spent to learn a new technology or process, which is type of waste and affects project performance (Lin et al. 1997).

Furthermore, this factor can play a major role for the completion of projects (Ozdamar and Alanya 2001; Tenenberg 2008) as well as errors rate, rework, and quality of the final product (Esaki et al. 2002; Alexander and Davis 1991).

3. **Requirements Clarity:** refers to the degree of clarity and comprehensive of the provided requirements to the development team (Rexfelt and Rosenblad 2006; Anton 2003; Jones 1996). In addition, the higher the understanding of customer's needs, the less rework, and re-changing to the requirements specifications; in which decreases project lead time, and effort wasted. Understanding the scope of the system is important to reduce effort in re-designing, reworking, and increasing quality of final product as lean suggest in both Sections 2.6 & 3.5.1 (Gotterbarn, 2002). In addition, the estimation errors decrease significantly with defined goals, and clear scope (Morgenshtern et al. 2007; Tiwana 2004; Maxwell and Forselius 2000). Requirements are the pre-stage to define the product's specification; hence their percentage of clarity is important for the proceeding phases.

4. **Requirements Volatility:** it is the frequency of changing, adding, and deleting the specification of the software product over the system development lifecycle (Zowghi and Nurmuliani 2002; Ferreira et al. 2009).

   It is an important control factor that can be an indicator of project's failure or success (Kulk and Verhoef 2008; Suardi 2004). Requirements change in late stages

of the software development lifecycle has a significant effect on any project performance measures, e.g. schedule, cost, and quality of the final product (Poppendieck and Poppendieck 2008).

Thus, changing user needs continuously can jeopardise the completion of the project (Maqsood and Javed 2007). Moreover, the quality of final product can be affected, e.g. stability and usability, as the level of requirement volatility increases (Jonsson and Wohlin 2006; Berander and Wohlin 2003). Kulk and Verhoef (2008) illustrated that requirement volatility rate depends on the size of the project and its duration. Hence, projects with shorter durations can accept a high rate of requirements volatility compared to long ones.

5. **Rework Rate:** it can be of three sources:
    i. Bugs which are the defects that injected into the final product through the lack of inspections.
    ii. Lack of experience, or human errors.
    iii. Introduction of new product or procedure.

According to Deephouse et al. (1995), reducing the percentage of rework has a significant positive effect on final project outcomes e.g. cost, schedule, and quality. Although, it is important to reduce this factor to increase the positive value of performance measurement (Pavur et al. 1999); complete avoidance of rework is often impossible (Liu et al. 2008). Flexibility of software process reduces rework significantly (Dekleva, 1992) and the research has adopted flexibility as shown in Table 2.1 by adopting the recommended practices and tools shown in Section 3.5 and Figure 3.4.

6. **Management Involvement and Control:** refers to the support and encouragement the organisation management provides to the development team. Providing proper training, high-tech tools, allocating experienced developers, rapid respond to management quires, and leadership have a positive impact on the project performance (Verner and Evanco 2005). On the other hand, poor management can significantly decrease flow, pace's speed, and block the horizontal communication channels that link the different departments the development has to pass through. Lean encourages the involvement of top management which can help in reducing any project risk factors in a timely manner (Whittaker 1999; Berander and Wohlin 2003; Dyba 2003; Reel 1999; Solingen 2004; Liu et al. 2008; Holmberg et al. 2008).

7. **Customer Involvement:** Close relationship with customer helps to shorten the feedback loop and speeds process of development (Whittaker, 1999). Lack of customer involvement is the cause of 40%-60% of software defects and failure (Anton, 2003). According to the Standish Group's 1995 CHAOS survey, lack of user involvement is in the top three 'project challenged' factors. Close relationships with customers helps in maintaining more successful profitable businesses (Ahmed and Capretz 2007; Petter 2008).

   Balancing this factor is important because of its negative influence as high customer involvement through the whole stages of project lifecycle will increase requirement volatility, thus negative effect on project performance (Berander and Wohlin 2003; Chow and Cao 2008; Enkel et al. 2005).

8. **Inspection Level:** Although testing is one of the software development lifecycle phases, its degree and existence in every workstation, or testing every possible input/output of system function play major role in software performance, lead time, quality, cost, etc. (Huang and Lo 2006). The fewer defects (bugs) produced, the less effort needed from quality assurance team to reject or correct (Hopp and Spearman 2000). Fewer bugs mean enough time to root cause the existing ones and enhance the process instead of the product (Chatzigeorgiou and Antoniadis 2003; Munson 1996).

   The sooner in the product lifecycle defects are discovered, the easier and less effort is needed to fix them. During development, rework costs 5-20% in relation to its detection location (testing, deployment, etc.) (Ebert and Man 2008).

9. **Availability of Developers:** refers to the absolute time the developer spends on the development process. Allocating the proper number of developers at the start of the project helps to reduce project lead time; whereas adding new developer in the middle of the process will require him/her sometime to be familiar with the project and its development environment (i.e. learning factor).

   Furthermore, having a developer working on more than one project will require the developer to re-adjust when shifting between them. This action is being considered to be a type of LSD Waste, 'task switching' in Section 3.5.2. Maxwell and Forselius (2000) considered this factor to be a critical factor for productivity. They link capacity load in software operations to be reduced by the effort needed, in which providing more time of each developer on the development process.

10. **Communication Overhead**: is related with the timing of getting the information through the organisation hierarchy, e.g. from the team leader to the developer located on the shop floor. Other researchers (Sawyer and Guinan 1998) called this factor 'social processes'. It encompasses internal group coordination activities (e.g. discussion of better solutions, missing design features, updates, etc.), solving conflicts of which design or tools should be used, etc. This factor is considered by many software researchers and practitioners to be a productivity determinant (Tausworthe 1992; Lin et al. 1997; Wu 2006). The allocation of tasks and synchronisation are both affected by communication (Bolton and Dewatripont 1995). Communication overhead is a function of team size (Lin et al. 1997).

Previous factors are the core of the research in the sense they reflect the analysis to literature, and formal/informal interviews. They reflect the reality with its uncertainty and variability in operating any service project. These generic operational/software factors will be modelled and used as part of the proposed method to plan and manage the software project.

3.10 Risk Management in Service Operations

Risk is a potential future problem that not yet happened and it would affect the success and the lead time of the project (Dey et al. 2007; Wallace and Keil 2004; Bennett et al. 1996). Risk is considered to be part and parcel of projects and covers all aspects of organisational activities (Tchankova 2002; Barros et al. 2004; Huang and Han 2008; Pang et al. 2006; Williams et al. 2006). In literature, risk managements methods agree on a standard steps include (Nguyen et al. 2010):

a. Risk identification.
b. Risk evaluation and quantification.
c. Risk classification.
d. Risk treatments to minimise its impact and continuous monitoring throughout project course work.

The highly dynamic and complexity of service projects, as governed by the characteristics presented in Table 3.1, necessitates the need for risk analysis and assessment (Cohen and Palmer 2004; Ropponen and Lyytinen 2000; Boehm 1989). Different approaches are being implemented to analyse, mitigate, tackle and avoid risk factors, e.g. McFarlan's portfolio and Boehm's software risk approach (Lyytinen et al. 1998). Additionally, risk factors should be considered during the plan and control stage to increase project success chances (Costa et al. 2007).

3.10.1 Definition of Software Risk

Software Risk Management has been defined by Ropponen and Lyytinen (2000) as "*an attempt to formalise risk oriented correlates of success into a readily applicable set of principles and practices*". It is the probability of the undesirable event multiplied by the value (usually in currency amount) of impact or magnitude of loss when that event occurs (Boehm, 1991).

Treating risk factors took different paths as well. Some forms deal with risks after they occurred to contain their damage as in crises management (McConnell, 1996); others tempted to prevent its causes. The research here will focus on the latter type, i.e. prevention and elimination of root causes of risks as this strategy aligns with lean

philosophy in eliminating of waste or causes of waste presented in Sections 2.6 and 3.5.1. Furthermore, it is easier, less costly, and more efficient to avoid risk factors in the first place, rather than solving, saving, and fixing problems after their occur (Cervone, 2006).

3.10.2 Existing Software Risk Management Techniques

There are different techniques, both qualitative and quantitative, in dealing with software risks such as expert interviews, expected monetary value, response metrics, Monte-Carlo methods, and Failure Mode and Effects Analysis (FMEA) (Carbone and Tippett 2004). Dey et al. (1994) applied Analytical Hierarchy Process (AHP) to have a systematic approach in a subjectivity judgement where risk analysis for contingency allocation is accomplished.

Accordingly, the following tools are some examples of the approaches identified through literature reviews and practices to carry out software risk management (Bannerman 2008; Wallace and Keil 2004; Burgess et al. 2001; Guthrie and Parikh 2004; Feather 2004; Bennett et al. 1996).

1. **Checklists**: it is a list that contains the high impact/top risk factors that can affect success of the project. For example, Boehm (1991) introduced the ‘top 10 risk identification checklist’ based on a survey of several experienced project managers to help in identifying the potential of those risks in their project. However, the list is not sufficient to identify present risk factors in relation with the rapid growth in complexity and technology of software products.

Ropponen and Lyytinen (2000) identified software risk components. They collected data set based on a survey sent to the Finnish Information Processing Association and the results being analysed through principal component analysis (PCA). Ropponen and Lyytinen (2000) have identified six software risk components:

a. scheduling and timing,
b. system functionality,
c. subcontracting,
d. requirements management,
e. resource usage and performance, and
f. personnel management.

However, their research was based on project managers self-report and the collected data is limited to certain time and field.

2. **Analytical Frameworks**: it is a non-process based framework where risk factors are clustered into categories according to related feature, such as their logical source, project lifecycle and socio-technical model with tasks, actors, structure, technology and their inter-relationships are elements of the model (Tchankova 2002; Keil et al. 1998; Wallace and Keil 2004). In such categorisation, control measures and monitors activities are easily applied to the whole group instead of individual risk factors (Bannerman 2008; Caldwell 2008). An example is Han and Huang (2007) analytical model where they used an empirical study that is based on 115 software projects on analysing the probability of occurrence and impact on 27 software risks. They divided the 27 risks to 6 dimensional groups:

a. user,
b. requirement,
c. project complexity,
d. planning & control, and
e. team and organisational environment.

The ‘requirement risk dimensional’ is the principal factor that has largest effect on software projects (Han and Huang 2007). However, their research did not consider the effect of software project attributes such as type, size, duration and staffing on the six dimensional software risk.

3. **Process Models**: it is a stepwise process that specifies activities in handling risk factors. These activities are:
    a. risk identification,
    b. risk analysis,
    c. risk response,
    d. risk control, and
    e. risk monitor.

    These activities can follow different sequences than mentioned earlier where suggestions of using some tools or techniques in one of those activities is abdicable (Bannerman 2008; Donaldson and Siegel 2007). An example is Dey et al. (2007) model as it considers risk management from developers’ side perspective based on both quantitative and qualitative techniques. Their model is integrated with the software development lifecycle and involves stakeholders concerns to redirect the development based on the framework feedback about risk factors.

4. **Risk Response Strategies**: it is a reaction activity to risk impact based on project circumstances, magnitude of threat, cost of response and resources required for the response in order to eliminate, reduce or mitigate it.

   There are four common risk response strategies found in literature (Bannerman 2008; Williams et al. 2006);

   a. **avoidance or terminate,** to eliminate the negative impact of risk occurring by many ways such as changing project design, adding more resources, etc.,
   b. **transference**, where passing the responsibility of handling the risk to a third party i.e. insurance, outsource contractor and Warranties Company,
   c. **mitigation/treat**, where reinforcing actions are designed to reduce risk root causes or lowering its effect on project success i.e. using test-driven approaches (Mateosian, 2003), and
   d. **acceptance**, they are of two types, passive and active. Passive response is when risk exists but do nothing except monitor its effects, since threat is low and source of risk is external. Active response is when there is little can be done to risk and a contingency plan is made to contain it when risk occurs.

Habitually the above four approaches are consistent and often used mutually to improve outcomes of the software project (Bannerman, 2008). However, through literatures, the following factors have played a major role in limiting the highly practices of risk management (Williams et al. 2006, Yetman 2004);

   a. risk is abstract and fuzzy,
   b. risk methods are mostly based on quantification of risks analysis,

c. risk impact has different implications toward stakeholders, and

d. risk tools are complex and costly to use.

As a result of the above draw backs, the research is embedding risk management within the planning and control process. Using risk checklist is the type of risk assessment forms the research here is applying in the proposed method. It will be used to highlight and get attention of management team to events that might occur and can cause risk or failure in the project.

Chapter 4: Research Methodology

4.1 Introduction

This chapter illustrates the steps undertaken to build the research methodology. As mentioned earlier in Section 3.2, service system has its unique characteristics that can create variability in the operation process. Identifying these factors and testing their effect on the promised goals help in meeting the customer requirements along the different phases of the project.

The main aim of the current research is to develop an operational management tool that will introduce proposed steps to help in planning and control small, medium and large software projects.

The research objectives are to achieve the above aim through four main tasks:

1. Briefly, identify software factors that may affect the rate and the time of completing different activities and tasks of the project.
2. Collecting and validating the required data to develop the research method.
3. Adopting QFD concept in planning and control projects.
4. Integration of simulation model with Design of Experiment (DOE) and Genetic Algorithm (GA) to plan, manage and control every phase of the project to meet customer demands.

The research here used software project environment as a case study for managing projects in service operation sector. However, the proposed method can be used in any service and manufacturing sectors as will be explained in Section 6.4. The tool

concentrates not only in fulfilling customer orders or demands, but highlights risk factors that may affect the completion of the project.

4.2 Overview of Research Design & Methodology

Research methodology is the knowledge claims and assumptions that are investigated through adopting strategies of inquiries. Such Strategies provide specific directions for procedures in a research design and analyse models. According to literature review (Creswell 2003; Ljungberg 2002); research methodologies are mainly classified in two types, Quantitative and Qualitative strategies.

4.2.1 Quantitative Methodology

This type is widely used in the social science and natural sciences e.g. physics, biology, sociology, etc. (Antonioni et al. 2007; Camargo 2001). It is based on the use of mathematical models, theories and hypotheses in an iterative process where evidences are evaluated and refined (Bertrand and Fransoo 2002). Quantitative Methodology focuses on objective rather than subjective and consists of the following steps (Khalil et al. 2010);a. collecting the required data using survey techniques, experiment techniques where conditioned experiment applied to a random sample, and quasi-experiment technique to study a specified sample,b. examining the collected data to identify the problems using given mathematical model or hypothesis,c. applying statistical analysis to identify the relationships in the considered data, and d. visualising results in form of tables and charts.

4.2.2 Qualitative Methodology

This method is used to investigate in-depth the understanding of human behaviour and reasons governing these behaviours through ‘why’ and ‘how’ not just ‘when’ and ‘where’ (Tetnowski and Damico 2001; Tunali and Batmaz 2003; Apaiah et al. 2005). It uses multiple interactive and humanistic approaches. It emphasises in choosing small and focused sample not random as in the previous method. Qualitative Methodology focuses on subjective (descriptive as data are in words, or pictures) rather than objective (e.g. numbers).

It consists of the following steps (William and Pearce 2006):

a. Collecting required data through interviews, focus groups, case study, direct observation, and action research where researcher participates in the process under the study (Hines et al. 1997).
b. Fundamental interpretation and describing the collected data.
c. More interpretation of the data, processes, entities and interactions observed.
d. Examining the focused sample to identify the problems in it.

4.2.3 Triangulation

This type is a combination of both quantitative and qualitative techniques. It includes predetermined and emerging methods, open and close-ended questions, statistical and text analysis (Creswell, 2003). This type may convert qualitative themes and codes into quantitative number to do a comparison analysis.

4.3 Research Methodology

The current research has adopted both qualitative and quantitative research methodology to help in the data collection. The validation and verification of every step is carried out using a case study from ADNOC Company in United Arab Emirates. Through literature and interviews, the research briefly identified ten factors that can affect project completion time as presented in Section 3.9. However these factors can be generic and changed according to the nature of the operation i.e. service and manufacturing environments.

The research methodology in this research had taken the following paths to gather needed data for analysis:

a. Collecting the needed data through literature reviews, formal and informal interviews, to identify the software factors and to model the software project.
b. Generating data using discrete event simulation to visualise the impacts of different software factors (i.e. being identified in previous step) affecting the software project and to test different scenarios.

Shortly, the main approaches and techniques adopted in this research are given.

4.3.1 Discrete Event Simulation Model

Since 1950s, project management are joined with use of network-based techniques such as Critical Path Method (CPM) and Programme Evaluation Review Technique (PERT) Project scheduling/rescheduling to help overcome the uncertainty that exists in earlier stages of the project (Ahuja and Thiruvengadam 2004). However, there are limitations in working with these tools that need to be overcome as presented in Section 6.5.

Project management needs the aid of decision-making tools to help converting the available information into actions. Such decision-making is controlled as well by wide explicit and implicit policies of behaviour. Therefore the research here has used Simulation as a tool to map the process of the different phases of the project. Simulation is used as well to act as an integrating tool for illustrating the effect of the different factors that affect the completion of a project with different performance indicators to meet customer order. Its suitability, appropriateness, and relevance are major factors to spread its use in practical real-world-applications, e.g. in the field of operations management (Jahangirian et al. 2010). Simulation here was integrated with design of experiments to find out which factor has a greatest effect on performance measurements Through literature review, two types of simulation methods are widely used as analysis tools; dynamic simulation, and Discrete Event Simulation (DES) (Raffo and Wernick 2001; Pfahl and Lesbsanft 2000).

The consideration of the service project as ‘n’ distinct stages with different timing distributions (e.g. exponential distribution) had pushed the research here to adopt DES as the simulation tool to investigate the effects of the identified factors on performance measurements (Berkely 1996; Erlang 1917). Moreover, DES provides the service of control the operations management, looking forward/backward in project planning, understand the internal/external relations of the performance variables (Kouskouras and Georgiou 2007; Raffo and Wernick 2001), and the ease in identifying the levels of blocking, waiting, and stoppages within workstations of the process (Khalil, 2006).

Discrete event simulation (DES) is a technique to model a continuous process occurring in the real world and breaking it into potential events (Chwif et al. 2006; Wohlgemuth et al. 2006; Robinson 2002; Taylor 1999). Therefore, the research here has used DES not only as a tool to simulate project, but as mentioned before will be integrated with DOE as a structured set of experiments.

Therefore, this research has adopted Simul8, which is a computer-based modelling package, as DES to model the different software projects sizes. Simul8 package supports the functions needed to proceed with the current research, such as;

a. creates visual models, input different values, import/export and from/to Excel,
b. allows the randomness to be modelled which enables the model to behave as actual systems would and overcome the limitation of getting exact measurements, and
c. the ease of gathering output results as the system group them in summaries.

4.3.2 Selection of Experimental Methodology (Design-of-Experiment)

Taguchi is a simplified method for Design-of-Experiment (DOE) used widely in manufacturing operations to optimise the processes in producing the products (Sukthomya and Tannock 2005). This DOE approach is developed by Genichi Taguchi (Taguchi, 1987) which combined statistical tools and methods in scientific experimenting (Wang and Huang 2008).

The use of Design of Experiments (DOE) enables obtaining the appropriate data by planning and designing of the needed experiments to be run. Moreover, DOE (i.e. using Taguchi technique) provides the ability to analyse the gathered data according to their

effect on a chosen response variables by the use of statistical methods to validate the conclusion (Antony et al. 2001).

Through literature reviews, few researchers had applied Taguchi's Design-of-Experiment in managing software projects. Tsai et al. (2003) have integrated OA with critical resource diagram (CRD) to allocate the appropriate human resources (developers) for the proper task. Salem et al. (2004) have applied Taguchi's OA to develop a logistic regression model. DOE helped in reducing the number of test cases needed to investigate the system quality. However, none of the previous models have applied Taguchi's OA to investigate the effects of the operations factors of a software project.

To reduce the cost of testing the many possible combinations of the parameters, Taguchi Orthogonal Arrays are used to determine the experiments to be carried out and the factors setting to be applied (Salem et al. 2004). It enables helpful and adequate information to be achieved from the designed experiments where reproducibility of the experiments is possible.

Taguchi technique is being applied in the current research to structure the experimental trials rather than the traditional Design-of-experiment use. It is an alternative to the traditional ineffective and undependable one-factor-at-a-time method to experimentation, where the analyst generally varies one factor or process parameter at a time keeping all other factors at a constant level. In traditional approach, all the possibilities are being investigated (equivalent to full factorial experiments) (Carino,

2006), e.g. in the case of this research (10 operational factors) with three levels will require only 27 trials of experiment compared with 59049 trials in the case of traditional DOE. Taguchi helps in testing the identified factors independently and gives different scenarios. The research here has integrated the Taguchi array with the simulation model. The experiment enabled the behaviour of the project operations activities to be understood in a short period of time and resulted in significantly improved performance (with the opportunity to design further experiments for possible greater improvements).

4.3.3 Applying Regression Analysis

Ramli et al. (2011) had defined Regression analysis to be "*a generic statistical tool to explore and describe dependencies among variables*". Regression analysis is used widely to help understand the behaviour of the dependent variables and the independent variables.

He et al. (2007) had applied fuzzy linear regression approach to identify the relationship between consumer satisfaction and productivity on service firm profitability. Regression analysis results had shown that firms can adopt information technologies to raise efficiency to achieve a balanced profitability and acceptable level of consumer satisfaction. Gunduz et al. (2005) had used regression approach to identify the major factors that may disturb the project output which the research here has investigated in more details.

This research used regression to identify which of the performance measurement indicators has significant effect on the completion of the project and its throughput.

4.3.4 Optimising the Factors

Optimisation is the process of finding a feasible solution according to a certain criteria (e.g. minimising setup time, cost, etc.) from a finite set of elements with a specified objective function (Kasperski and Zielinski 2010, Wang et al 2008). Multidimensional optimisation has many applications in different fields including science, engineering, economics, etc. where mathematical modelling is mostly used (Georgieva and Jordanov 2009). In most cases, there exist some variables that need to be changed or modified entitled 'decision variables' (APICS, 2008) to achieve the optimal solution for the given optimisation problem.

Wang et al. (2008) had applied nonlinear stochastic optimisation to maximise the expected profit with high demand uncertainty. Their model use stochastic programming-based genetic algorithm (SPGA) to compute a profitable capacity planning and task allocation plan.

Lancaster and Cheng (2008) had used genetic algorithm optimal project scheduling to meet customer requirements. The method aimed in choosing the optimum sequence between the project activities to lower the total duration or lead time. In addition, Sakalauskas and Felinskas (2006) had used optimisation algorithm for planning project which based on heuristic priority rules. The results showed significant improvement in meeting requirements of job priority relations, and different project management constraints.

Optimisation also has been used to provide the optimal solution for the allocation of available developers in a software project at the planning phase with different constraints e.g. schedule deadline, budget, and head-count (Barreto et al. 2008).

Optimisation with genetic algorithm is being applied in this research to provide the optimum solution to the critical software factors in order to achieve the promised customer demands.

4.4 Proposed Steps for Research Experiment

The research here has broken the experimental phase into sequence of proposed steps to achieve the objectives of the research. They are sequential in the sense that step 1 occurs first followed by step 2 and then step 3, etc. Figure 4.1 presents the research proposed steps.

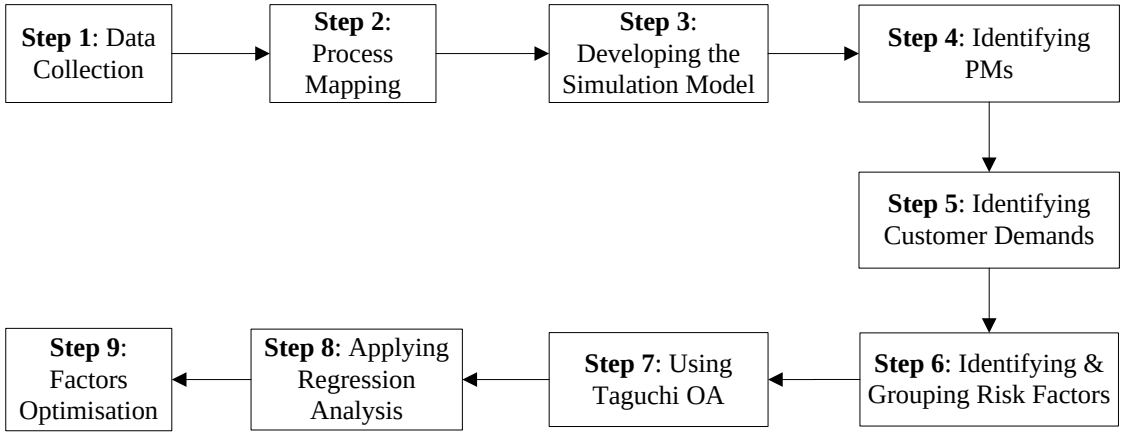

Figure 4.1: Proposed Research Steps

The steps are presented below:

Step 1: Data Collection

Understanding the operations occurring in software project is a primary step in identifying the required data needed to highlight the path for improvements. Accordingly, the needed data are gathered through two main methods:

i. **Historical data through literature reviews:** analysis to literature to find the different models of managing service projects is done. The research took the software project as a case study and literature review was used to verify each step undertaken in developing research method.

ii. **Interviews (formal and informal):** with project managers and team leaders of a local company in the United Arab Emirates, ADNOC.

ADNOC is a large company in UAE established in 1971 with a large IT department. They use different types of auto code generation tools. There is no standard software model the company follows rather than adopting pieces of existence SD models according to their needs. The size of software projects the company process suites the research investigation. The issues and concerns in such environment encouraged the research here to take part to gather the needed data to proceed with the investigation.

The interviews had taken place to gather the needed data about the parameters and measurements of each phase within the lifecycle of the software project. Such parameters data helped in developing the simulation model as presented in step 3. In addition, ADNOC IT team had validated the research proposed steps

and the three levels of the 10 operational factors to help in setting up Taguchi orthogonal arrays as will be shown in step 7.

Through the historical data that gathered in Step1-i, ten generic software project factors were identified and shown in Table 4.1. The research here used them to study their effect on the project activities. Their behaviour in the developed model is important to identify which of them have major effect on the project response variables. The ten factors were validated by ADNOC.

Table 4.1 presents the operational factors of software project.

Software Project Development Factors
1. Size of team
2. Requirements clarity
3. Requirements volatility
4. Reworks (bugs)
5. Management involvement
6. Customer involvement
7. Inspection level
8. Availability of developers
9. Skills
10. Communication overhead

Table 4.1: Operational (Software) Factors

Obtaining the accurate data that represents the behaviour of project activities is important task to identify the sources of wastes, risks, and variability. Accordingly, the parameters listed in Table 4.2 were added to the model.

1. Cycle time of each phase.
2. Different size of projects.
3. The needed resources (developers) for the project.
4. Different types of interruptions for each phase.
5. Mean-Time-to-Repair (MTTR) for each phase.
6. Mean-Time-to-Failure (MTTF) for each phase.

Table 4.2: Data Gathered through Interviews

Table 4.3 presents the types of Interruption for each phase that are gathered through the interviews in Step 1-ii. They will be input to the simulation model in the next step.

Modelling Elements	**Interruption**
Requirements Phase	Meeting Customers
	Team Meetings
	Learning New Stuff (e.g. UML, document formats)
Discussion Phase	Top Management Meetings
Design Phase	Team Meetings
	Changing Specification
Prototyping Phase	Meeting Customers
	Team Meetings
	Changing Design
Coding Phase	Team Meetings
	Learning new coding method
Testing & Integration Phase	Team Meetings
	Fixing Bugs
	Fixing Integration Errors
Deployment Phase	Training Customers

Table 4.3: Interruption Types

Interruptions are part of any project environment. Such interruptions affect the operations management and the flow through project lifecycle. Previous types of interruptions (i.e. stoppages) identified in Table 4.3 can be either:

a. long stoppages, where mean-time-to-repair (MTTR) is above 30 minutes, and
b. short stoppages, where mean-time-to-repair (MTTR) is less than 30 minutes.

Those two types of stoppages align with what is been mentioned in operations management literature reviews of having two types of events, planed events (e.g. monthly management meetings, maintenance, backup of data, etc.), and unplanned events (e.g. team short meetings, machine breakdowns, tool breakage, worker absenteeism, lack of material, scrap, rework, requirements changes, etc.) in which such events are most likely undesirable (Taylor, 1999).

Step 2: Process Mapping

Through Literature reviews in Section 3.4, different software development project phases exist. Figure 4.2 shows the different departments/phases the software project passes through for completion. The proposed method started by drawing a flow chart presented in Section 3.3.2 to reflect the flow of information and completed work i.e. material among software project management phases.

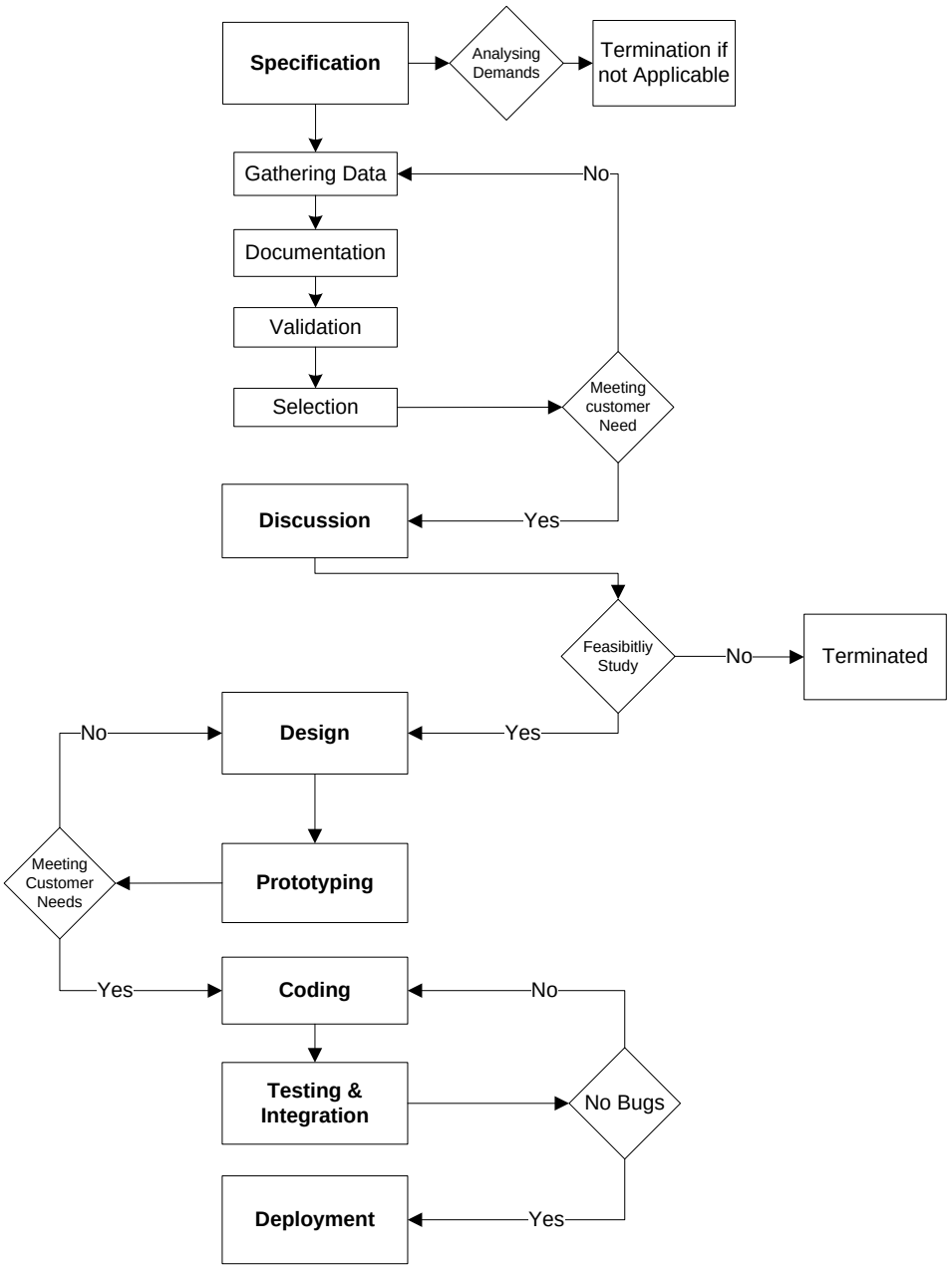

Figure 4.2: Process Mapping of the Software Project

Step 3: Developing the Simulation Model

The simulation model is developed according to the mapped process mentioned in Step 2. Each phase of the process is represented as a workstation in Simul8.

In addition, the data that gathered in the interviews (formal and informal) are used to build the simulation model to present different sizes of software development projects as listed in Table 4.4.

Simulation Parameters	Value
Results Collection Period	It represents different size of project 3months i.e. 28800 minutes, 6months i.e. 57600 minutes, and 9months i.e. 86400 minutes (small, medium, and large).
Travel Time	It was set to Zero, as the model represent a real life project and avoid the effect of any other factor, which may change final results.
Random Time	No randomness as discussed in travel time.
Shift Pattern	9am-5pm equivalent to 8hrs per day , 5 days per a week
Probability Distribution	Triangle distribution was chosen as a trade-off between accuracy (Khalil et al. 2010)
Resources	Multi task developers

Table 4.4: Simulation Parameters

Step 4: Identifying Performance Measurements

According to Kasunic (2008), Performance Measuring "*is a process of assessing the results of a company, organisation, project, or individual to determine how effective the operations are, and make changes to address performance gaps, shortfalls, and other problems*". Setting up measures can help in identifying and minimising the wastes presented in Section 3.5.2, but also can be used as monitor elements to fulfil customer order and reflect if there is any risk.

Moreover, performance measurements (PMs) are an operational management measures to highlight the value-added or non-value-added activities i.e. highlight the factors that can affect completion of project. Therefore, the collections of measurement data help

the improvement of such process, through analysis, interpretation, and identification of the strength and weaknesses (Ljungberg 2002; Barclay and Osei-Bryson 2010; Lima et al. 2009).

From literature reviews (Bhasin, 2008), the key measures to operations perspectives in the software project are been identified and listed in Table 4.5. These PMs are integrated in the Operations Project Management Deployment (OPMD) model and act as response variables.

Performance Measurement for Each Phase	**Unit of Measure**
1. %Waiting	Percentage
2. %Blocking	Percentage
3. %Stoppages	Percentage
4. %Working	Percentage
5. Throughput Rate	Modules/minute
6. Project Lead time	Minutes
7. Throughput	Modules

Table 4.5: Operation Project Management Deployment PMs

They add quantification sense to the process in which helps visualising sources of delays, rework, unnecessary processing, and resources utilisation. Performance measurements help measure the impact of enhancement actions on such mapped value stream (Alkaabi et al 2010).

Step 5: Identifying Customer Demands

Through literature reviews shown in Section 3.8, the current research had identified the customer demands presented in Table 4.6.

Customer Demands
1. Completed Modules: Completed tasks or modules
2. Deliverables: Produced modules or task per unit time
3. Delivery Time: Project duration which has been assumed to be 3, 6, and 9 months.

Table 4.6: OPMD Customer Demands

Customer demands will be the indicators of success from failure regarding project status. They are adopted to be the goals/targets the project seeks to achieve. Therefore, the optimum solution as in Step 9 will be given in regard to achieve customer demands/requirements.

Step 6: Identifying and Grouping Risk Factors

Through literature reviews, different types of risk techniques exist as mentioned in Section 3.10.2; the research here has chosen the checklist type. In the case of the difference performance measurements values shown in Table 4.5 are not meeting customer demands, OPMD will use the checklist to highlight that there is a risk level among the different project phases.

i. The research here has adopted the Generic Software Acquisition Management Project Risk Factors (2008) that are being developed through Texas Delivery Project. The list consists of 82 generic risk factors that explore risks entities within the software development environment.

ii. The clustering step is helpful in managing risks factors. It is meaningful to cluster risk factors into sets that contain risks related to the same area or similar. Through literature, different criteria for similarity are being identified (McFarlan

1981; Barki et al. 1993; Wallace et al. 2004; Ropponen and Lyytinen 2000), including:

a. Type of risk; technical, personnel, organisational, schedule, functionality, etc.
b. Criticality; highly critical, moderate, etc.
c. Stakeholder; customers, developers, analyst, etc.

McFarlan (1981) had grouped software risks into three categories, project size, project structure, and technology experience. Barki et al. (1993) identified five dimensions of software risk factors, system size, expertise, technological newness, system complexity, and organisational environment.

Accordingly, the research here has grouped the 82 risk according to their type into five groups. This categorisation provides a comprehensive consideration to all the aspects that are mentioned earlier along with the operations issues that can affect any factor of software development project success.

Step 7: Using Taguchi Orthogonal Array

Taguchi Orthogonal Array was selected to model the various combinations of the software factors identified in Step 1. The identified generic 10 software factors are levelled into three levels. The choice of three levels is related to the depth of analysis the current research is aiming at. Therefore, their impact on the response variables can be measured.

These levels were being validated through ADNOC Company. Some modifications are made to the levelling according to their recommendations and they are agreed to be as shown in Table 4.7.

Factor	Level 1	Level 2	Level 3
Size of team	2 developers	4 developers	6 developers
Requirements clarity	60% are clear	75% are clear	85% are clear
Requirements volatility	5% changed	10% changed	15% changed
Reworks (bugs)	5% rework	15% rework	25% rework
Management involvement	20% involvement	40% involvement	60% involvement
Customer involvement	30% involvement	60% involvement	90% involvement
Inspection level	at start of project	at the middle or end	at each workstation
Availability of developers	4 Hrs	8 Hrs	12 Hrs
Skills	65% of team are skilled	75% of team are skilled	85% of team are skilled
Communication overhead	10%	15%	20%

Table 4.7: Taguchi L27-a Orthogonal Array

Therefore, LA-27a was chosen from the available selection of orthogonal arrays. Results will be collected automatically from simulation which as well acts as a response variable in LA-27a, i.e. performance indicators for completing the project. Accordingly, the simulation will run 27 times to represent 27 experiments using the same random number stream. This action is not to lose accuracy by running different streams of values. Table 4.8 presents the orientation of the 27 experiments to be run in aim to test the different combinations of the ten generic software factors identified in Step 1.

Size of Team	Requirements Clarity	Requirements Volatility	Rework (Bugs)	Management Involvement	Customer Involvement	Inspection Level	Availability of Developers	Skills	Communication Overhead
2	60%	5%	5%	20%	30%	1	4	65%	10%
2	60%	5%	5%	40%	60%	2	8	75%	15%
2	60%	5%	5%	60%	90%	3	12	85%	20%
2	75%	10%	15%	20%	30%	1	8	75%	15%
2	75%	10%	15%	40%	60%	2	12	85%	20%
2	75%	10%	15%	60%	90%	3	4	65%	10%
2	85%	15%	25%	20%	30%	1	12	85%	20%
2	85%	15%	25%	40%	60%	2	4	0%	10%
2	85%	15%	25%	60%	90%	3	8	75%	15%
4	60%	10%	25%	20%	60%	3	4	75%	20%
4	60%	10%	25%	40%	90%	1	8	85%	10%
4	60%	10%	25%	60%	30%	2	12	65%	15%
4	75%	15%	5%	20%	60%	3	8	85%	10%
4	75%	15%	5%	40%	90%	1	12	65%	15%
4	75%	15%	5%	60%	30%	2	4	75%	20%
4	85%	5%	15%	20%	60%	3	12	65%	15%
4	85%	5%	15%	40%	90%	1	4	75%	20%
4	85%	5%	15%	60%	30%	2	8	85%	10%
6	60%	15%	15%	20%	90%	2	4	85%	15%
6	60%	15%	15%	40%	30%	3	8	65%	20%
6	60%	15%	15%	60%	60%	1	12	75%	10%
6	75%	5%	25%	20%	90%	2	8	65%	20%
6	75%	5%	25%	40%	30%	3	12	75%	10%
6	75%	5%	25%	60%	60%	1	4	85%	15%
6	85%	10%	5%	20%	90%	2	12	75%	10%
6	85%	10%	5%	40%	30%	3	4	85%	15%
6	85%	10%	5%	60%	60%	1	8	65%	20%

Table 4.8: Orientation of the Operational Factors Using Taguchi Orthogonal Array

Step 8: Applying Regression Analysis

When performance measurements results are not matching customer demands, risk level will be raised to highlight the risk factors or events that may cause the PMs to be behind customer demands. Regression analysis will help to identify the software factors with high impact on performance measurements. In this step, not all the 10 software factors will be optimised only the one with high impact. Regression is helpful because it can show a predictive relationship that can be subjugated in practice (Duellmann et al. 2010).

The research then will use DOE to identify the critical software factors or variables affecting the identified PMs through use of Regression analysis. This action is important to reduce the variables list to a manageable size.

Step 9: Factors Optimisation

After the identification of the most correlated operational factors, the need comes to provide the optimum values/solution for these factors in order to achieve customer demands. Such optimum values can clarify the uncertainty for project manager when planning, and what action he/she needs to be taken to achieve project goals.

Chapter 5: Experimental Results

5.1 Introduction

This chapter demonstrates the experimentations results and some observations after running the designed experiments according to the methodology developed in the previous chapter. The main aim of the current research is to develop an operational management tool that will help in planning and managing different size projects. This aim is achieved through implementing the research methodology steps described in chapter 4.

The main result of the conducted experiments showed that %waiting significantly affects project completion time and its throughput rate. Analysis to the variables behind the %waiting allowed the identification of two critical software factors that they play major role in maximising it. The experimentations showed that requirement volatility and communication overhead which are being modelled as Mean-Time-To-Repair/Mean-Time-To-Failure and Queuing time are these two variables that can be optimised. Moreover, the results after optimisation showed improvements in the chosen performance measurements. Such improvements can push the plan to suite the customer demands and make the project succeed.

5.2 Results of the Research Experiment Proposed Steps

Step 1: Data Collection

As mentioned in Section 4.3, the undertaken research has used mixed method of the both quantitative and qualitative techniques in aim to gather the needed data. Such

data is used to develop the simulation model of software project as will be shown in Step 3.

In modelling the software project operations, the research has identified ten software factors/variables that affect project activities as shown in Table 4.1 and explained in Section 3.9. Modelling some of these variables in Simul8 took different form as these variables cannot be represented in its actual form.

Table 5.1 presents the modelling of the ten software factors in the simulation model.

Input Factors/Variables	Modelling Property	Explanation
1. Size of Team	Resources	Identify number of developers involved in every phase.
2. Requirements Clarity	MTTR/MTTF	The clearer the requirements, the easier and faster to start the development; therefore, the effect of this variable is being represented through Mean-time-to-Repair (MTTR)/Mean-time-to-Failure (MTTF). Each work station (phase) will be paused until the needed requirement is being clear and understood.
3. Requirements Volatility	MTTR/MTTF	The effect of requirements volatility is to stop the work after occurring. This variable is being reformed through modelling element to be Mean-to-time-Repair/Mean-time-to-Failure. In addition, the triangle distribution function will be used to feed up the values of MTTR/MTTF as a trade-off between accuracy.
4. Rework (Bugs)	%Rework	The tasks that will be rework on as cause of faulty or bug and will be modelled with the specific function (Rework) available in Simul8.
5. Management Involvement & control	Setup Time	Management will always assume time added up for attending meetings, brainstorming, workshops, approvals, etc. It is the added up which is not included in the task time. It will be fed as a Value.
6. Customer Involvement	Processing Time	This variable can have both positive impact on shortening the processing time (in upstream stations/phases) or negative impact on increasing processing time (in downstream stations/phases). Modelling this variable will be through its effect on the operations time of each workstation.
7. Inspection Level	%Rework	According to test, work can be redone, redesign. Inspection should be on every phase as implementing lean. In Simul8, this variable is modelled through %rework which will be a value added.
8. Availability of Developer	Allocation of shift pattern	It is the shift or working period of each developer/resource through the project period. This is modelled through specifying the shift pattern to be either, 4hrs, 8hrs, or 12 hrs.
9. Skills	Resource	Move the operator or not according to skill.
10. Communication	Queuing Time	It is the flow of information upon receiving the order from

overhead		the customer, MRP to supplier (customer) to different departments that include e-mails, meetings, scheduling, job allocation, etc. This variable is reformed in the modelling process to be part of the Queuing Time.

Table 5.1: Operational Variables Identification

Steps 2 & 3: Process Mapping and Developing the Simulation Model

The identified phases/stages in Figure 4.1 of the project lifecycle have been built as workstations in Simul8. Each workstation represents a phase/stage of the software project lifecycle, where the default attributes' values of the workstation is modified to emulate the intended phase based on the data presented in Table 5.2.

Moreover, the following attributes are used to build the simulation model. They are been collected and verified through interviews with ADNOC Company team.

Table 5.2 presents the values for previous mentioned attributes.

Modelling Elements	**Value**		
Resources	Multi task developers		
Requirements Phase	Cycle Time (min)	3400, 3800, 4200	
		MTTR (min)	**MTTF (min)**
	Meeting Customers	10, 20, 30	1200, 1900, 3800
	Team Meetings	15, 30, 45	1200, 1900, 3800
	Learning New Stuff (e.g. UML, document formats)	25, 35, 45	1200, 1900, 3800
Discussion Phase	Cycle Time (min)	860, 960, 1060	
		MTTR (min)	**MTTF (min)**
		10, 20, 30	320, 480, 960
Design Phase	Cycle Time (min)	2000, 2400, 2800	
		MTTR (min)	**MTTF (min)**
	Team Meetings	15, 30, 45	800, 1200, 2400
	Changing Specification	40, 50, 60	800, 1200, 2400
Prototyping Phase	Cycle Time (min)	2000, 2400, 2800	
		MTTR (min)	**MTTF (min)**
	Meeting Customers	10, 20, 30	800, 1200, 2400
	Team Meetings	15, 30, 45	600, 800, 1200
	Changing Design	40, 50, 60	800, 1200, 2400
Coding Phase	Cycle Time (min)	10000, 12000, 14000	
		MTTR (min)	**MTTF (min)**
	Team Meetings	15, 30, 45	4000, 6000, 12000
	Learning new coding method	10, 15, 20	3000, 4000, 6000
Testing & Integration Phase	Cycle Time (min)	4400, 4800, 5200	
		MTTR (min)	**MTTF (min)**
	Team Meetings	15, 30, 45	1600, 2400, 4800
	Fixing Bugs	10, 15, 20	960, 1200, 1600
	Fixing Integration Errors	40, 50, 60	1600, 2400, 4800

Deployment Phase	Cycle Time (min)	2000, 2400, 2800	
		MTTR (min)	**MTTF (min)**
	Training Customers	30, 40, 50	800, 1200, 2400

Table 5.2: Simulation Modelling Elements Attributes

Steps 4 &5: Identifying Performance Measurements and Customer Demands

Figure 5.1 presents the integration of the identified performance measurements in Section 4.4 Step 4, customer demands in Section 4.4 Step 5, along with the ten generic software factors.

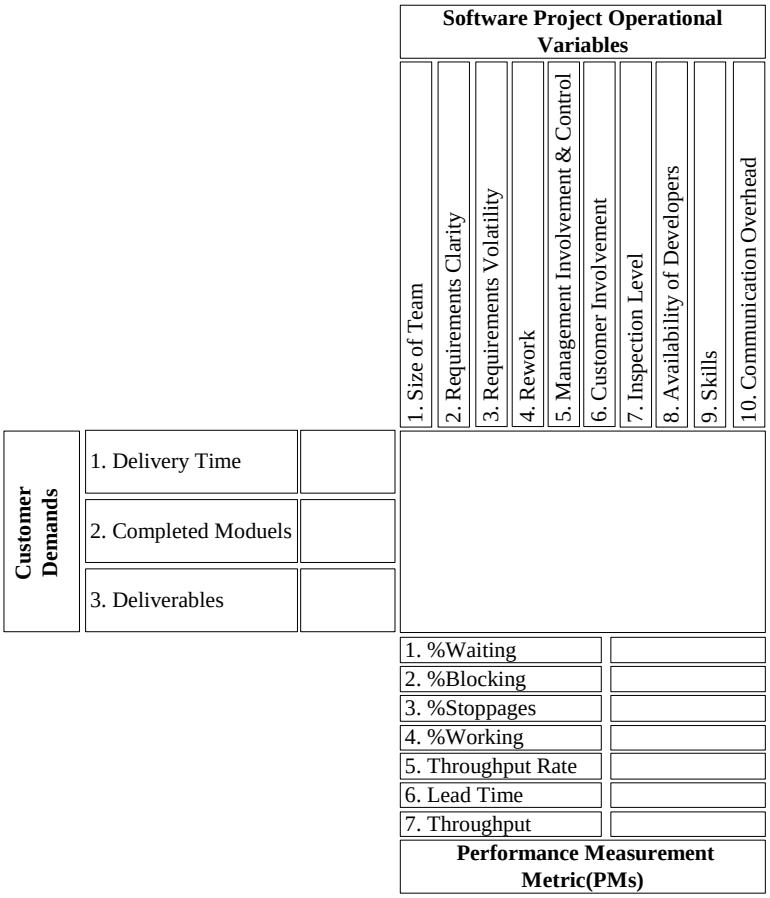

Figure 5.1: Integrating Customer Demands, PMs, and Software Factors

Step 6: Identifying and Grouping Risk Factors

The 82 software risk factors are grouped into five groups as shown in Table 5.3. They are integrated in the OPMD model to highlight risk events.

Software Risk Category	Software Risk Factors
1. Preparation Risks	• Project Fit to Customer Organisation. • Project Fit to Provider Organisation. • Customer Perception. • Work Flow. • Goals Conflict. • Resource Conflict. • Customer Conflict. • Leadership. • Program Manager Experience. • Definition of the Program. • Political Influences. • Convenient Date. • Attractive Technology. • Short Term Solution.
2. Organisation Risks	• Organisation Stability. • Organisation Roles and Responsibilities. • Policies and Standards. • Management Support. • Executive Involvement. • Project Objectives. • User Involvement. • User Experience. • User Acceptance. • User Training Needs. • User Justification.
3. Project Structure Risks	• Project Size. • Hardware Constraints. • Reusable Components. • Supplied Components. • Budget Size. • Budget Constraints. • Cost Controls. • Delivery Commitment. • Development Schedule. • Requirements Stability. • Requirements Complete and Clear. • Testability. • Design Difficulty. • Implementation Difficulty. • System Dependencies. • Hardware Resources for Deliverables. • Response or Other Performance Factors. • Customer Service Impact. • Data Migration Required. • Pilot Approach. • External Hardware or Software Interfaces.
4. Development Process	• Alternatives Analysis. • Commitment Process. • Quality Assurance Approach. • Development Documentation. • Use of Defined Engineering Process. • Early Identification of Defects. • Defect Tracking. • Change Control for Work Products. • Physical Facilities. • Hardware Platform. • Tools Availability. • Vendor Support. • Contract Fit.

	• Disaster Recovery.
5. Personnel Risks	• PM Approach. • PM Communication. • PM Experience. • PM Attitude. • PM Authority. • Support of the PM. • Team Member Availability. • Mix of Team Skills. • Application Experience. • Experience with Project Hardware and Software. • Experience with Process. • Training of Team. • Team Spirit and Attitude. • Team Productivity. • Expertise with Application Area (Domain). • Technology Match to Project. • Technology Experience of Project Team. • Availability of Technology Expertise. • Maturity of Technology. • Design Complexity. • Support Personnel. • Vendor Support.

Table 5.3: Software Development Risk Factors Structure

Figure 5.2 below illustrates OPMD after adding the risk factors groups.

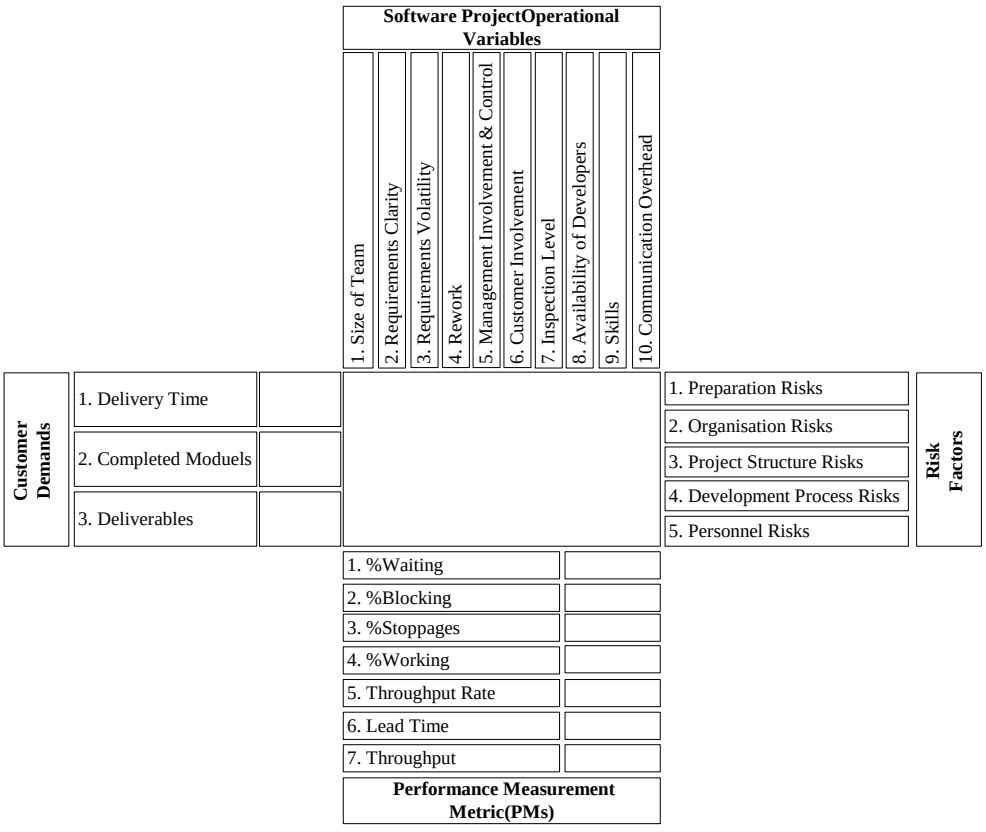

Figure 5.2: The Complete Components of OPMD

Step 7: Results of Taguchi Orthogonal Array

The results of the response variables (PMs) are been collected for the software project after running 27 experiments according to the suggested combinations in Table 4.8. In addition, three sets of experiments are been conducted regarding three different size projects, 3months (i.e. 28800 minutes), 6months (i.e. 57600 minutes), and 9months (i.e. 86400 minutes).

The results of performance measurements before optimisation are being presented below in Tables 5.4-5.6.

a. 3months size project i.e. 28800 minutes:

Average %Waiting	Average %Blocking	Average %Stoppages	Average %Working	Throughput Rate (Modules/Minute)	Lead Time (Minutes)	Throughput (Modules)
8.08	19.40	14.24	58.27	230	238020	16286
10.02	19.42	16.75	53.81	238	238420	15467
5.97	15.86	16.22	61.94	224	237890	16630
5.07	17.35	14.13	63.44	230	237840	17400
3.94	17.52	11.72	66.81	223	237700	17950
9.25	19.28	15.37	56.10	237	238320	16120
4.14	15.60	17.28	62.98	225	237890	16980
3.16	15.97	16.53	64.35	231	237990	17520
8.09	23.09	11.25	57.57	253	237965	17520
5.64	16.64	16.83	60.89	245	237685	17770
5.82	18.65	10.90	64.63	264	237849	20436
8.62	22.43	10.58	58.36	274	237925	19486
5.86	17.57	12.92	63.65	278	237800	21086
6.29	20.54	9.01	64.15	279	237980	21636
5.22	16.08	14.00	64.70	298	237920	23001
8.13	20.24	17.44	54.19	298	237920	19738
5.06	18.10	7.95	68.89	288	237655	23896
7.26	21.43	9.91	61.39	303	237840	22631
5.74	17.53	12.91	63.82	303	237796	23340
4.65	14.36	13.46	67.53	291	237676	23340
3.76	20.25	10.66	65.33	370	237740	29037
5.31	19.87	15.23	59.58	271	237920	20074
6.28	18.18	16.66	58.89	345	237856	24650
4.72	17.05	9.48	68.75	368	237720	29987
4.85	17.51	12.61	65.03	272	237830	21024
5.03	17.71	9.14	68.12	320	237820	25900
6.43	19.33	8.79	65.44	305	237755	23990

Table 5.4: Performance Measurements Results for 3months size project

The following figure represents the %utilisation and %non-utilisation for the 3months size project before optimisation:

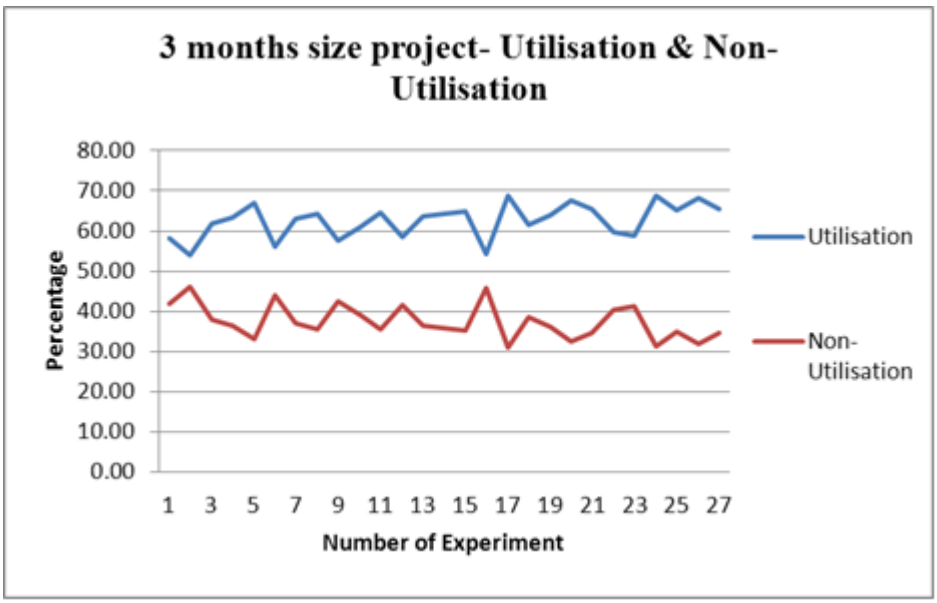

Figure 5.3: %Utilisation & %Non-Utilisation for 3months size project

b. 6months size project i.e. 57600 minutes:

Average %Waiting	Average %Blocking	Average %Stoppages	Average %Working	Throughput Rate (Modules/Minute)	Lead Time (Minutes)	Throughput (Modules)
10.91	26.19	19.23	43.66	229.79	321327.00	21986.10
10.02	26.21	22.61	41.16	238.30	321867.00	20880.45
5.97	21.42	21.90	50.71	224.10	321151.50	22450.50
5.07	23.43	19.08	52.42	230.47	321084.00	23490.00
3.94	23.66	15.82	56.57	223.19	320895.00	24232.50
9.25	26.03	20.76	43.97	237.17	321732.00	21762.00
4.14	21.06	23.33	51.47	224.86	321151.50	22923.00
3.16	23.15	23.96	49.72	231.21	345085.50	25404.00
8.09	31.17	15.19	45.55	253.49	321252.75	23652.00
5.64	27.46	27.76	39.13	244.58	392180.25	29320.50
5.82	25.18	14.71	54.29	263.64	321096.15	27587.93
8.62	32.52	15.35	43.51	274.26	344991.25	28253.98
5.86	28.99	21.32	43.83	277.55	392370.00	34791.08
6.29	27.73	12.17	53.81	278.97	321273.00	29207.93
5.22	24.92	21.70	48.16	297.70	368776.00	35650.78
8.13	27.32	23.54	41.01	298.02	321192.00	26645.63
5.06	28.05	12.32	54.56	288.27	368365.25	37038.03
7.26	28.94	13.38	50.42	303.37	321084.00	30551.18
5.74	23.67	17.43	53.16	303.27	321024.60	31508.33
4.65	20.82	19.52	55.01	291.27	344630.20	33842.28
3.76	27.33	14.39	54.51	370.49	320949.00	39199.78
5.31	32.79	25.13	36.77	271.38	392568.00	33122.10
6.28	24.54	22.49	46.70	345.49	321105.60	33277.50
4.72	26.43	14.69	54.16	368.43	368466.00	46479.66
4.85	23.64	17.03	54.48	272.28	321070.50	28382.40
5.03	23.90	12.34	58.72	320.03	321057.00	34965.00
6.43	26.10	11.87	55.60	305.22	320969.25	32385.83

Table 5.5: Performance Measurements Results for 6months size project

Figure 5.4 represents the %utilisation and %non-utilisation for the 6months size project before optimisation:

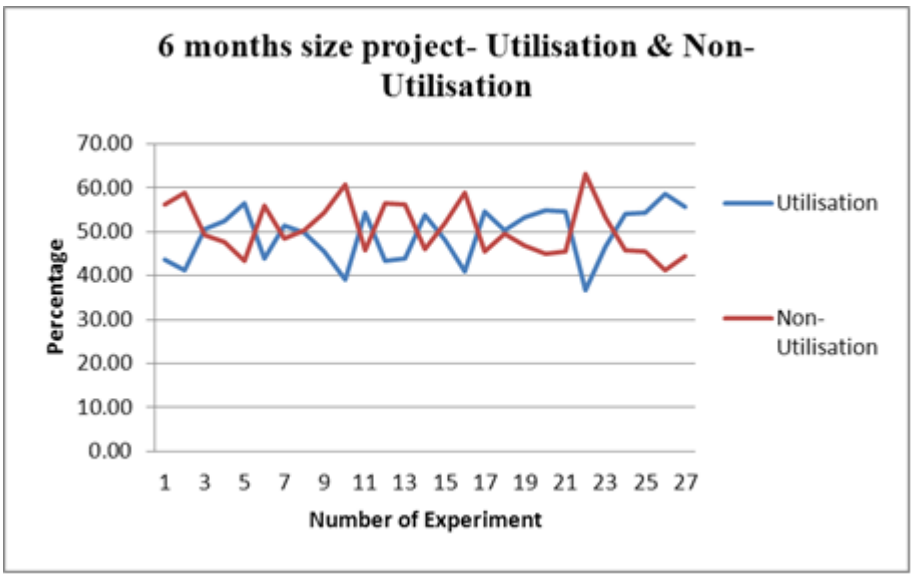

Figure 5.4: %Utilisation & %Non-Utilisation for 6months size project

c. 9months size project i.e. 86400 minutes:

Average %Waiting	Average %Blocking	Average %Stoppages	Average %Working	Throughput Rate (Modules/Minute)	Lead Time (Minutes)	Throughput (Modules)
11.72	28.13	20.65	33.03	229.79	345129.00	23614.70
14.53	28.15	24.29	44.81	238.30	345709.00	22427.15
8.66	23.00	23.53	43.34	224.10	344940.50	24113.50
7.86	26.90	21.91	51.87	230.47	368652.00	26970.00
5.72	25.41	17.00	36.35	223.19	344665.00	26027.50
13.41	27.95	22.29	46.32	237.17	345564.00	23374.00
6.00	22.62	25.06	48.30	224.86	344940.50	24621.00
4.58	23.15	23.96	38.47	231.21	345085.50	25404.00
11.73	33.48	16.31	35.08	253.49	345049.25	25404.00
9.37	27.62	27.93	48.72	244.58	394557.10	29498.20
8.44	27.05	15.80	39.63	263.64	344881.05	29631.48
12.50	32.52	15.35	47.29	274.26	344991.25	28253.98
8.50	25.47	18.74	48.02	277.55	344810.00	30573.98
9.12	29.79	13.07	52.35	278.97	345071.00	31371.48
7.05	21.71	18.90	33.58	297.70	321192.00	31050.68
11.79	29.34	25.28	54.89	298.02	344984.00	28619.38
7.34	26.24	11.53	44.02	288.27	344599.75	34648.48
10.53	31.08	14.37	43.92	303.37	344868.00	32814.23
8.89	27.18	20.01	52.91	303.27	368583.80	36176.23
6.74	20.82	19.52	49.73	291.27	344630.20	33842.28
5.46	29.36	15.46	41.40	370.49	344723.00	42103.47
7.70	28.82	22.09	40.39	271.38	344984.00	29107.30
9.11	26.35	24.15	54.69	345.49	344891.20	35742.50
6.84	24.73	13.74	48.07	368.43	344694.00	43480.97
7.20	26.01	18.73	53.77	272.28	353177.55	31220.64
7.29	25.67	13.26	48.17	320.03	344839.00	37555.00
9.65	29.00	13.19	78.60	305.22	356632.50	35984.25

Table 5.6: Performance Measurements Results for 9months size project

The following Figure 5.5 represents the %utilisation and %non-utilisation for the 9months size project before optimisation:

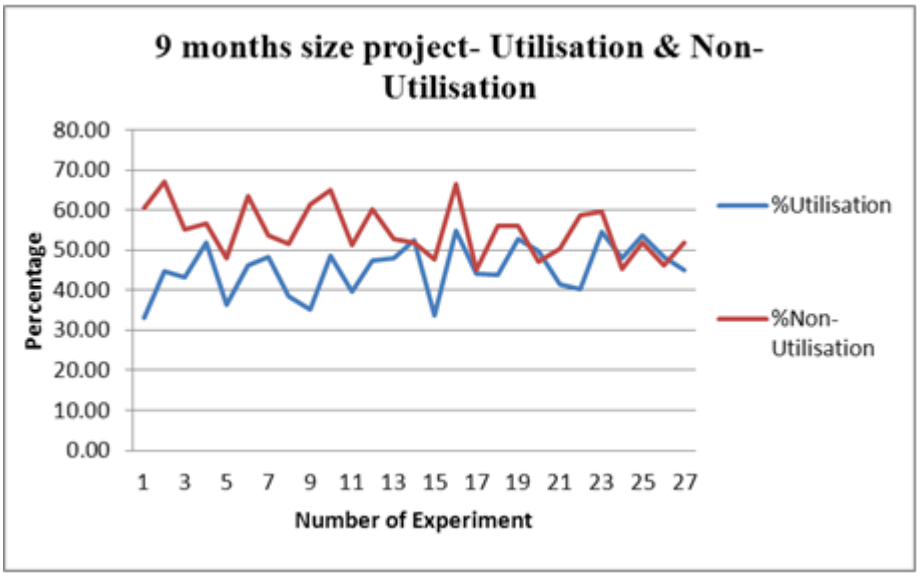

Figure 5.5: %Utilisation & %Non-Utilisation for 9months size project

The adoption of OPMD will aim to increase the %utilisation against the %non-utilisation as shown in previous Figures 5.3-5.5.

Adopting the visual management recommendation suggested by Toyota 14 ways in Section 2.7 allowed having the results integrated within OPMD structure. Figure 5.6 reflects the complete components of the OPMD with the results for one scenario:

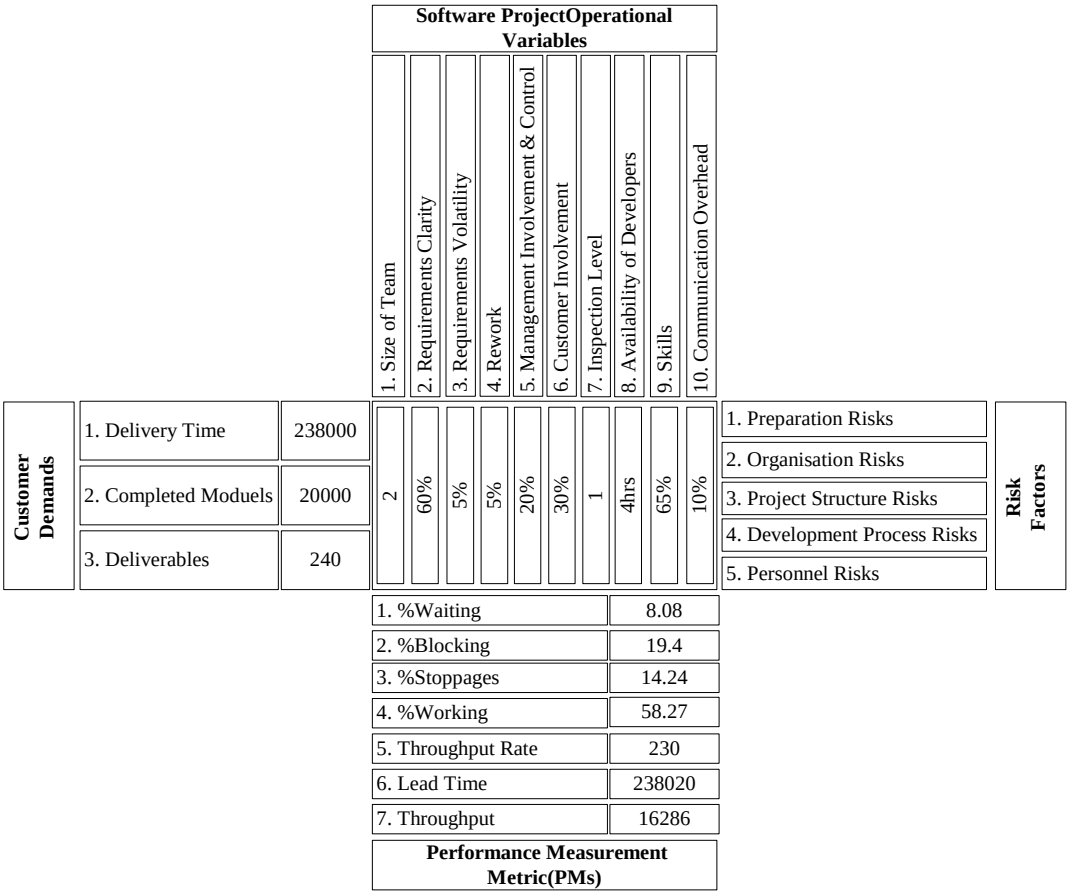

Figure 5.6: OPMD with results after running one scenario

Step 8: Regression Analysis

Regression analysis is being applied to identify the strength of relationship between each of the performance measurements i.e. %waiting, %blocking, %stoppages, and %working on the project completion time and project throughput, and Throughput Rate. Tables 5.7-5.9 and Figures 5.7-5.9 present the values of R-Square after applying the regression analysis to the simulation outputs. It is clear from these results that %waiting has highest R-Square on the project completion time, throughput, and throughput rate.

Throughput

Name	R-Square
% Waiting	0.65
% Blocking	0.55
% Stoppages	0.27

Table 5.7: Throughput R-Square Value

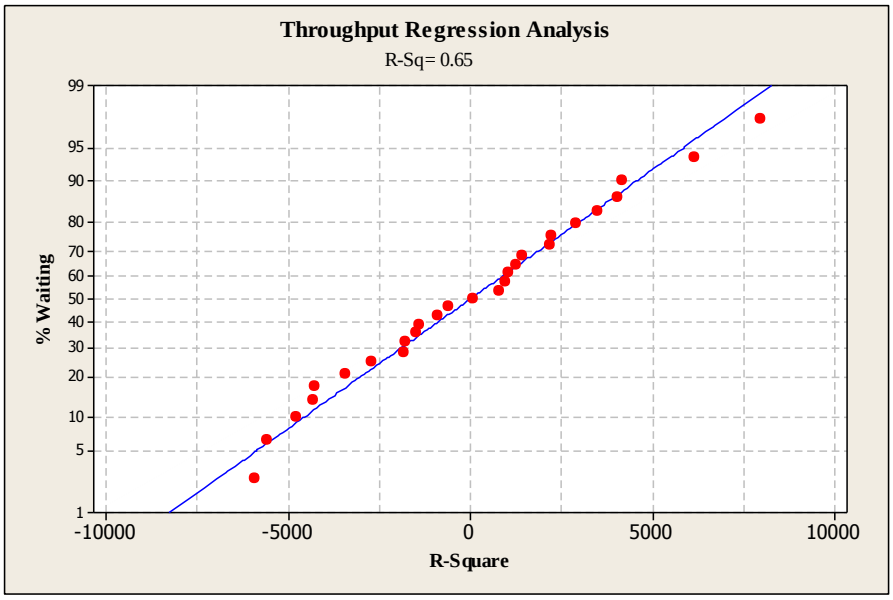

Figure 5.7: Throughput R-Square

Throughput Rate

Name	R-Square
% Waiting	0.85
% Blocking	0.50
% Stoppages	0.47

Table 5.8: Throughput Rate R-Square Value

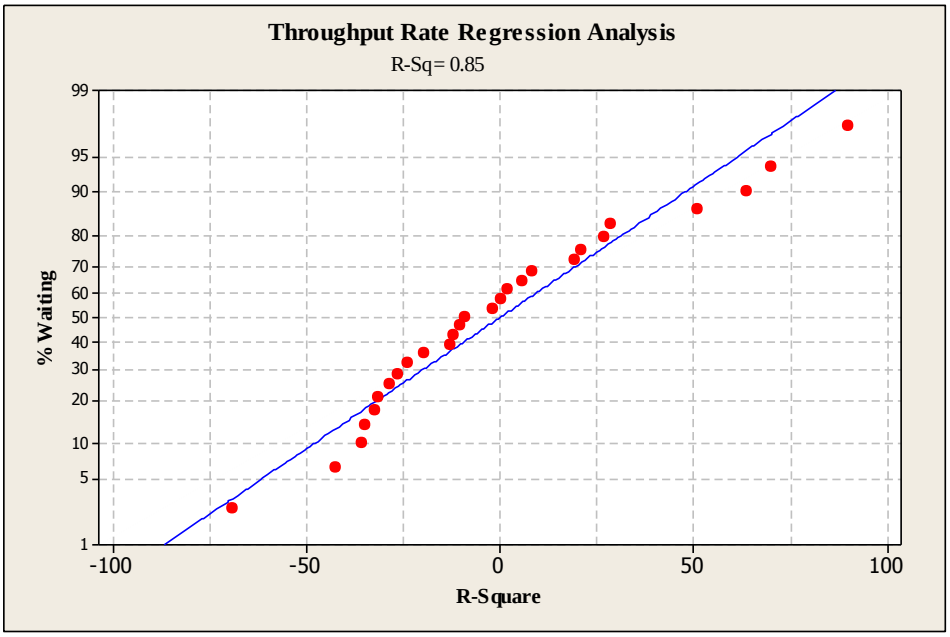

Figure 5.8: Throughput Rate R-Square

Lead time

Name	R-Square
% Waiting	0.90
% Blocking	0.52
% Stoppages	0.07

Table 5.9: Lead time R-Square Value

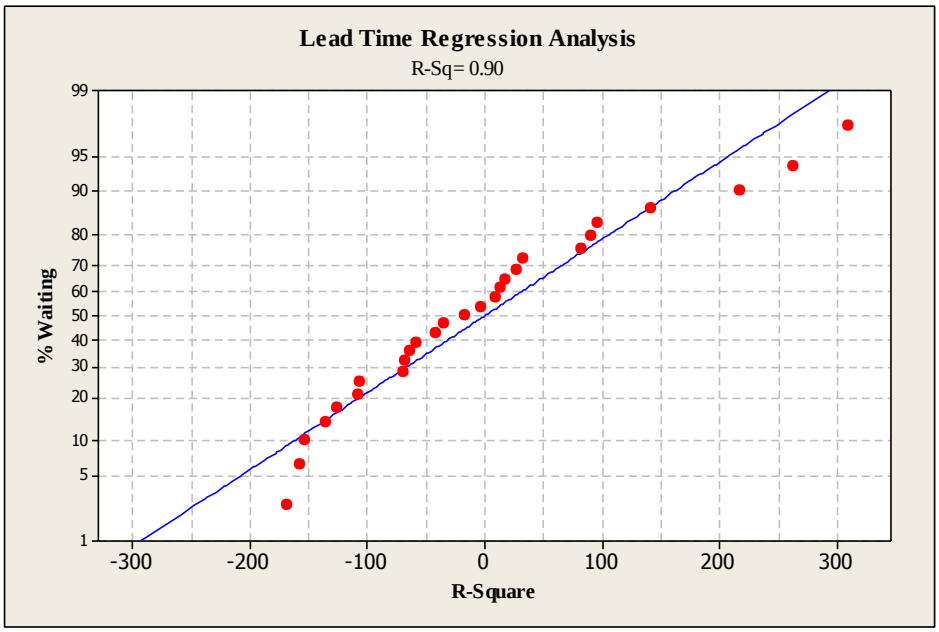

Figure 5.9: Lead Time R-Square

By looking at the results, it is been found that %waiting has the most significant influence on the performance measurement. Therefore, the research analysed Taguchi Orthogonal Arrays to identify the variables that can affect %waiting. It is important to reduce the variables list (i.e. identified in step 2) into a manageable size.

Figures 5.10 below shows the analysis of running DOE (Taguchi Orthogonal Arrays) on 3months size project:

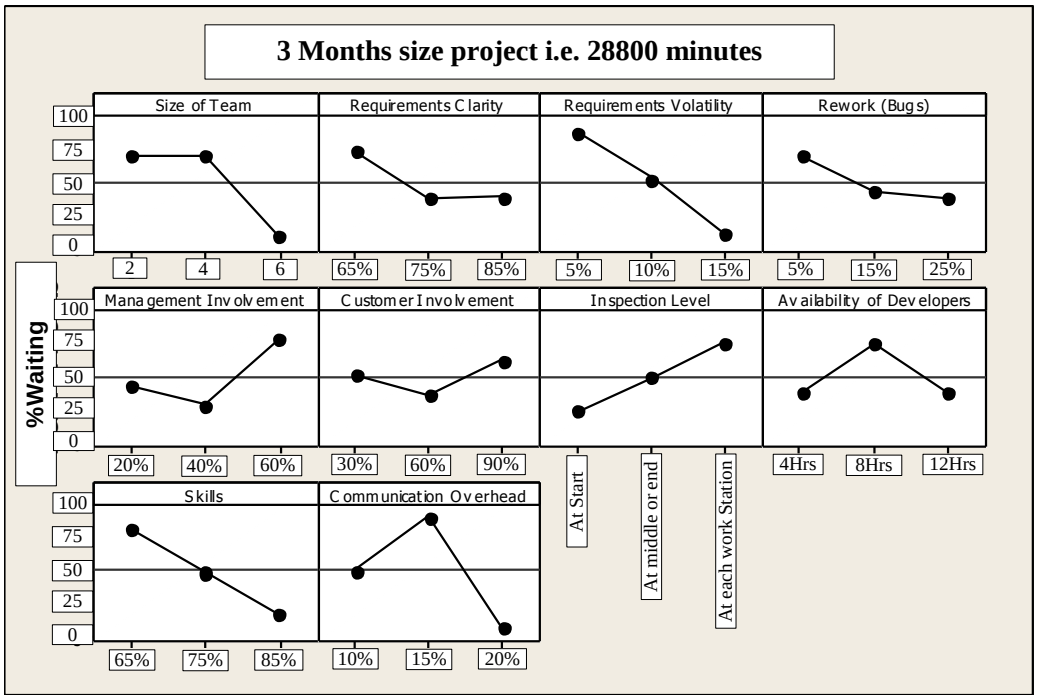

Figure 5.10: %Waiting Taguchi Analysis for 3months size project

Figures 5.11 below shows the analysis of running DOE (Taguchi Orthogonal Arrays) on 6 months size project:

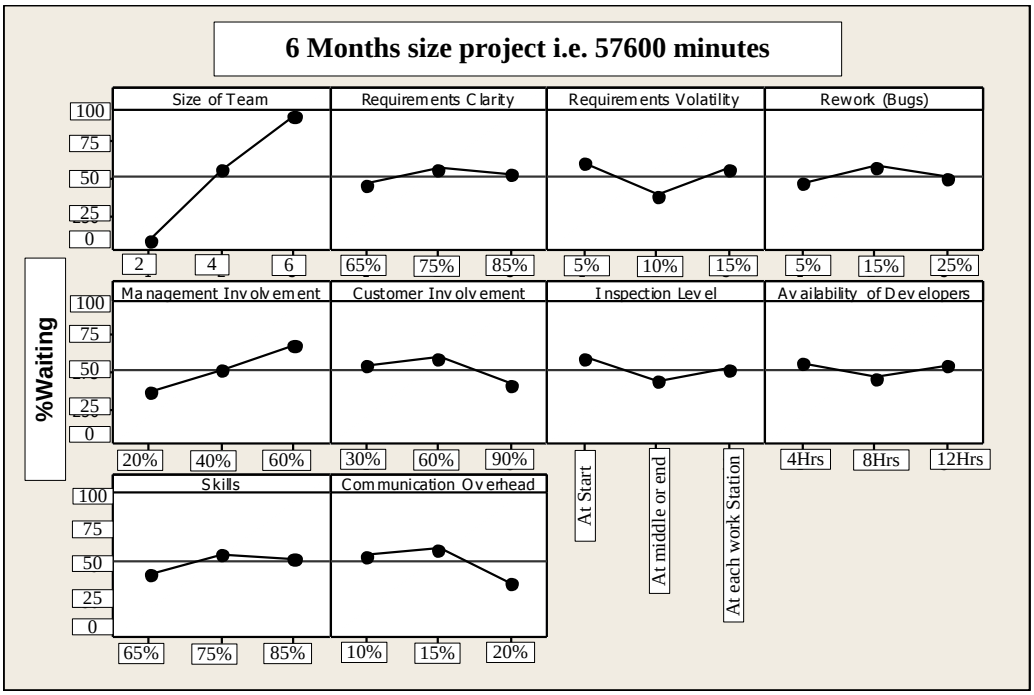

Figure 5.11: %Waiting Taguchi Analysis for 6months size project

Figures 5.12 below shows the analysis of running DOE (Taguchi Orthogonal Arrays) on 9months size project:

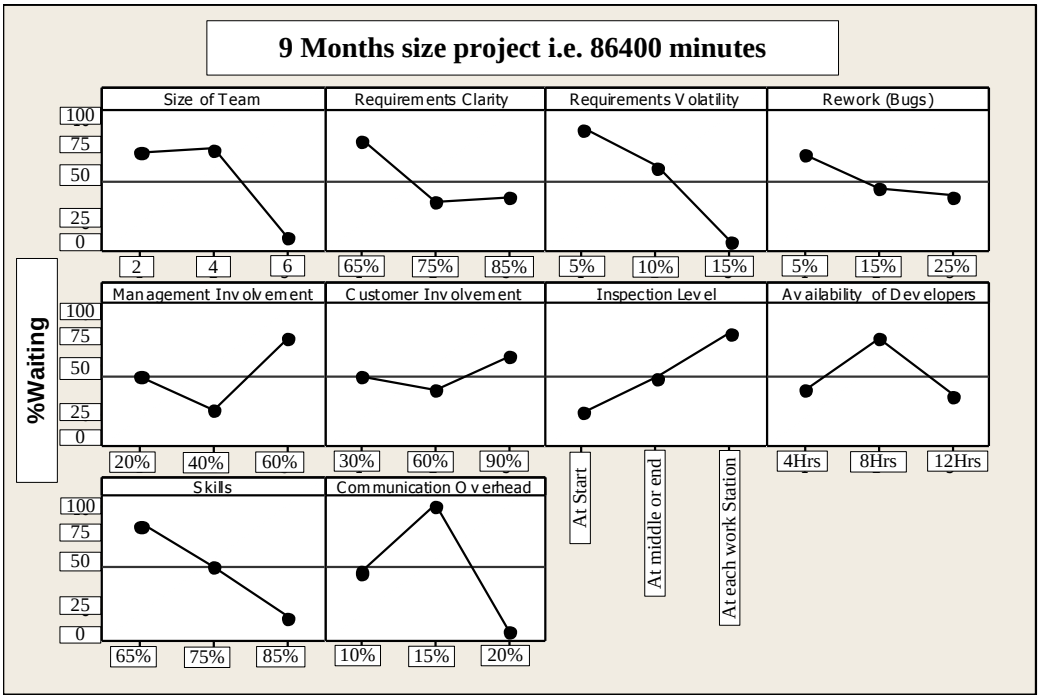

Figure 5.12: %Waiting Taguchi Analysis for 9months size project

Analysis of DOE illustrated that there exists two software factors (i.e. out of the ten generic software factors listed in Table 5.1) affecting %Waiting, they are:

a. **Requirements volatility**, i.e. its modelling attribute is Mean-time-to-Repair (MTTR)/Mean-time-to-Failure (MTTF), and

b. **Communication overhead,** i.e. its modelling attribute is Queuing Time.

Step 9: Applying Optimisation

Accordingly, the experimentation is proceeded by optimising the identified factors in Step 8 and re-running the simulation to visualise whether there are improvements in the performance measurements or not. Therefore, the research here re-runs the experimentation with three different periods (3months, 6months, and 9months) similar to Step 7 to compare the results with previous results in Table 5.4-5.6. Having three

sizes of projects will help to generalise the results and make the method applicable to any size of project.

The optimisations were conducted in two steps; start with optimising Mean-time-to-Repair/Mean-time-to-Failure, and then adding Queuing Time optimisation.

I. 3months Size Project:

The following Tables 5.10-5.11 present the results of performance measurements after optimising the Mean-Time-to-Repair/Mean-Time-To-Failure (MTTR/MTTF) and Queuing Time (QT) for 3months project i.e. 28800 minutes.

a. After MTTR/MTTF Optimisation only:

Average %Waiting	Average %Blocking	Average %Stoppages	Average %Working	Throughput Rate (Modules/Minute)	Lead Time (Minutes)	Throughput (Modules)
4.16	16.92	12.69	66.24	236.68	212029.89	28188.61
6.06	17.64	16.07	60.22	245.45	211558.55	30308.18
3.09	14.66	15.36	66.89	230.83	211514.09	31711.50
2.89	15.06	12.54	69.52	237.39	211389.58	32713.88
1.67	15.78	9.28	73.26	229.88	211940.96	31446.09
5.63	17.18	14.22	62.98	244.29	211558.55	30946.05
2.08	14.54	16.79	66.59	231.60	227325.07	34295.40
1.95	14.97	18.42	64.66	238.14	211625.25	31930.20
5.38	19.91	8.65	66.07	261.09	258348.74	39582.68
4.07	15.83	23.22	56.88	251.92	211522.09	37243.70
3.67	16.90	8.17	71.26	271.55	227262.99	38142.87
5.94	19.58	8.81	65.67	282.49	258473.74	46967.95
3.67	15.79	14.78	65.76	285.88	211638.59	39430.70
3.16	18.01	5.63	73.21	287.34	242931.19	48128.55
2.55	15.10	15.16	67.20	306.63	211585.23	35971.59
5.22	17.65	20.00	57.14	306.96	242660.61	50001.33
2.33	16.04	5.78	75.85	296.92	211514.09	47354.32
4.69	18.62	6.84	69.85	312.47	211474.96	42536.24
2.76	15.90	10.89	70.45	312.37	227025.14	45687.07
1.82	14.02	12.98	71.17	300.00	211425.15	52919.70
1.55	17.69	7.85	72.91	381.60	258604.17	44714.84
3.34	17.05	20.59	59.03	279.52	211528.31	44924.63
4.37	17.07	15.95	62.61	355.86	242726.98	62747.54
3.24	16.33	8.15	72.28	379.49	211505.19	38316.24
3.07	16.50	10.49	69.94	280.45	211496.30	47202.75
2.48	15.82	5.80	75.90	329.63	211438.49	43720.86
3.49	17.25	5.33	73.93	314.37	624607.20	23989.50

Table 5.10: 3months Project after MTTR/MTTF Optimisation

b. After MTTR/MTTF and Queuing Time Optimisation:

Average %Waiting	Average %Blocking	Average %Stoppages	Average %Working	Throughput Rate (Modules/Minute)	Lead Time (Minutes)	Throughput (Modules)
3.52	16.09	9.04	71.35	241.62	190826.90	38054.62
4.59	16.71	12.42	66.27	250.58	190402.70	40916.04
2.55	14.47	11.71	71.27	235.64	190362.68	42810.53
2.17	14.94	8.89	74.00	242.34	190250.62	44163.73
0.89	15.29	5.63	78.18	234.68	190746.86	42452.22
2.73	16.52	10.57	70.19	249.39	190402.70	41777.17
1.81	14.33	13.14	70.72	236.44	204592.57	46298.79
1.47	14.97	14.77	68.78	243.12	190462.72	43105.77
4.47	18.30	5.00	72.24	266.54	232513.87	53436.61
3.11	15.41	19.57	61.90	257.17	190369.88	50278.99
2.09	16.15	4.52	77.25	277.22	204536.69	51492.87
2.88	17.95	5.16	74.01	288.39	232626.36	63406.73
2.75	15.24	11.13	70.88	291.84	190474.73	53231.44
2.68	17.21	1.98	78.13	293.34	218638.07	64973.54
2.10	15.03	11.51	71.36	313.03	190426.71	48561.65
2.72	16.38	17.35	63.56	313.37	218394.55	67501.80
0.96	15.17	2.13	81.74	303.12	190362.68	63928.33
3.37	16.64	3.19	76.79	318.99	190327.46	57423.92
2.24	15.22	7.24	75.30	318.89	204322.63	61677.55
1.29	14.11	9.33	75.27	306.27	190282.64	71441.60
0.72	16.24	4.20	78.85	389.57	232743.75	60365.03
1.48	16.34	16.94	65.24	285.36	190375.48	60648.24
3.31	16.35	12.30	68.04	363.29	218454.28	84709.17
2.59	15.54	4.50	77.36	387.41	190354.67	51726.92
2.39	16.13	6.84	74.64	286.30	190346.67	63723.71
1.60	15.68	2.15	80.57	336.51	190294.64	59023.17
1.89	15.89	3.68	78.54	320.93	562146.48	32385.83

Table 5.11: 3months Project after MTTR/MTTF and Queuing Time Optimisation

Figures 5.13-5.19 present plots that compare the results for each of the identified performance measurements before Optimisation and after MTTR/MTTF and Queuing Time optimisation.

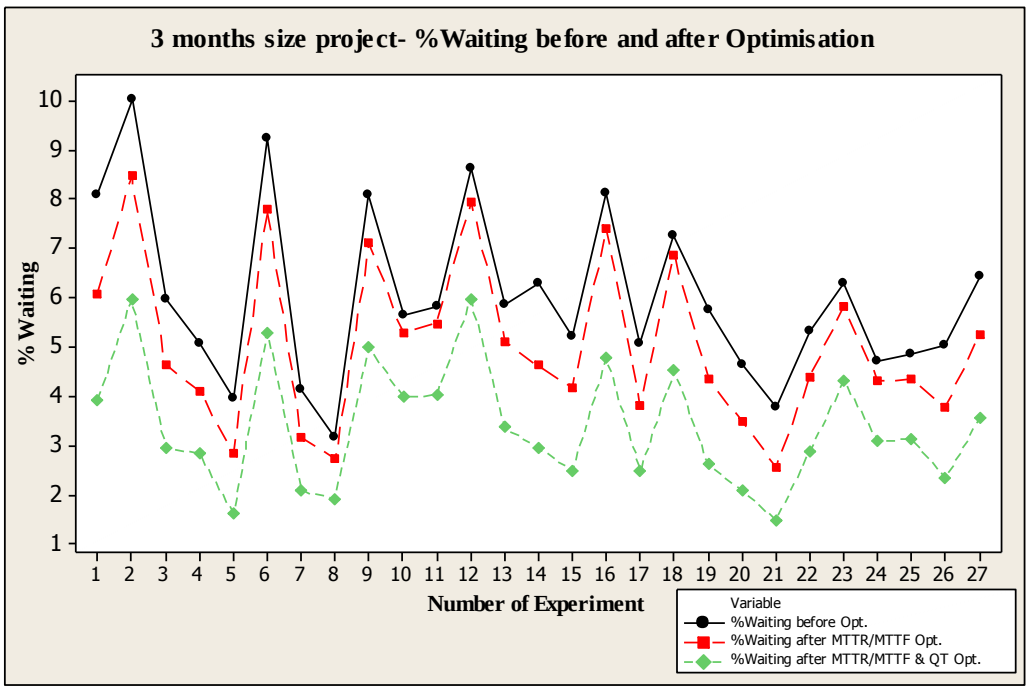

Figure 5.13: 3months Size Project %Waiting Before and after Optimisation

The figure illustrates a reduction in %waiting, e.g. reduced from 10% to 6% in experiment#2.

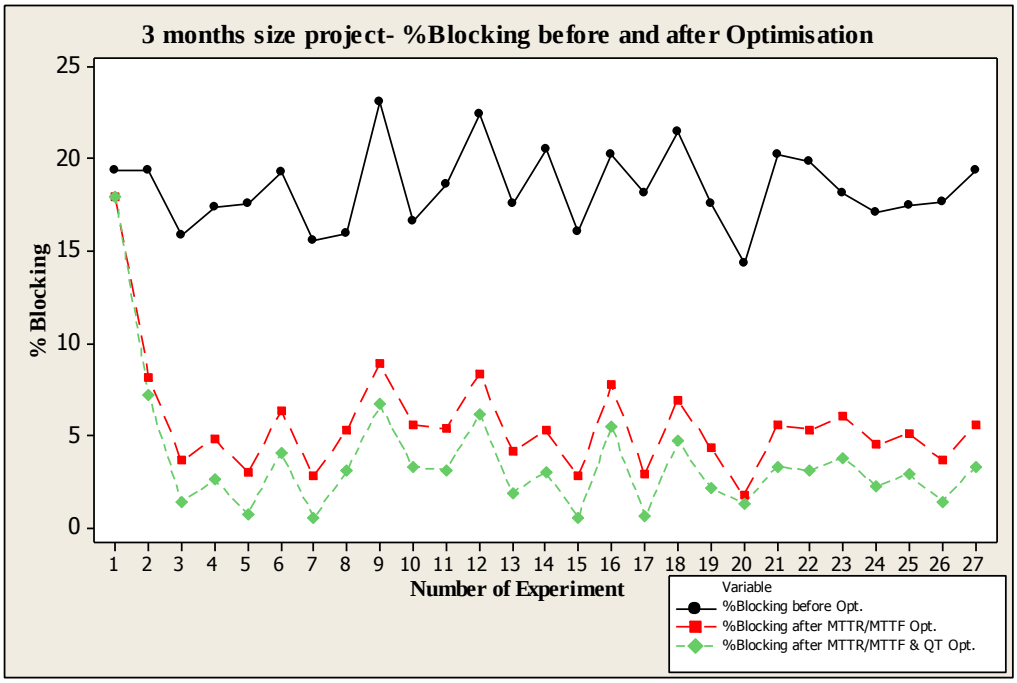

Figure 5.14: 3months Size Project %Blocking Before and after Optimisation

%blocking is reduced as well after optimising MTTR/MTTF and Queuing time. For example, Experiment#9 illustrated a drop from 24% to 5%.

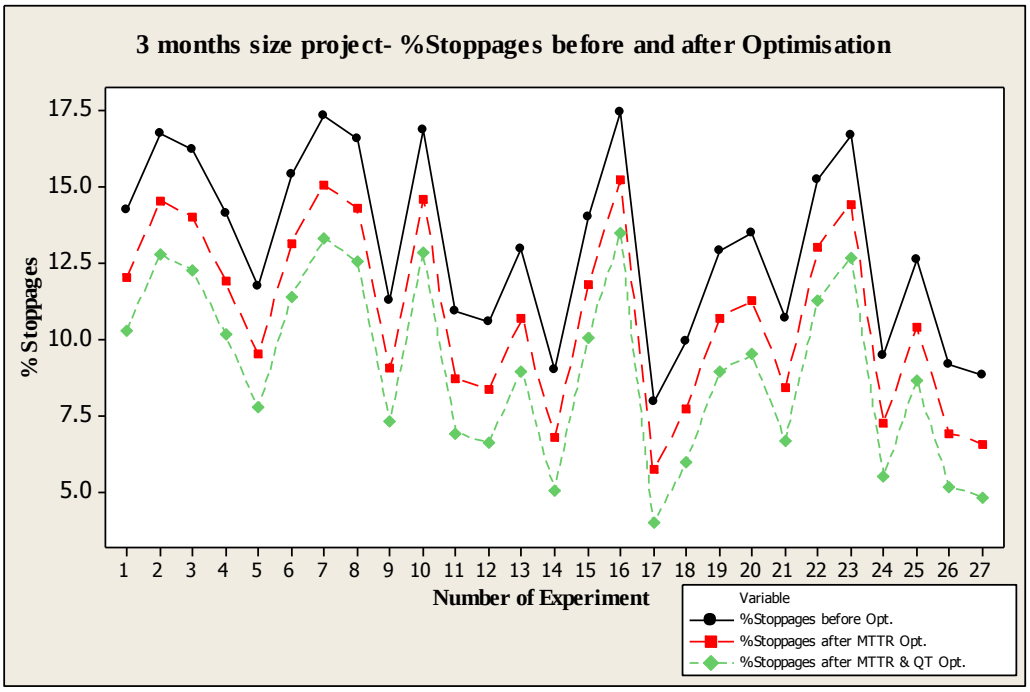

Figure 5.15: 3months Size Project %Stoppages Before and after Optimisation

Figure 5.13 illustrates the reduction in %stoppages after optimisation.

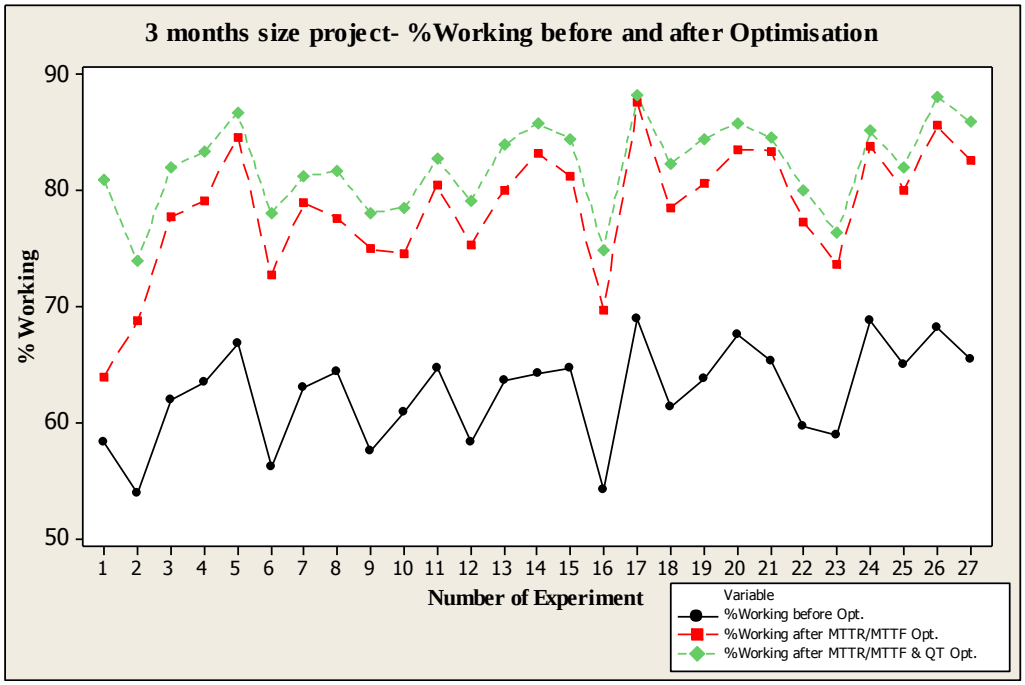

Figure 5.16: 3Months Size Project %Working Before and after Optimisation

%working has increased significantly regarding MTTR/MTTF and Queuing time optimisation.

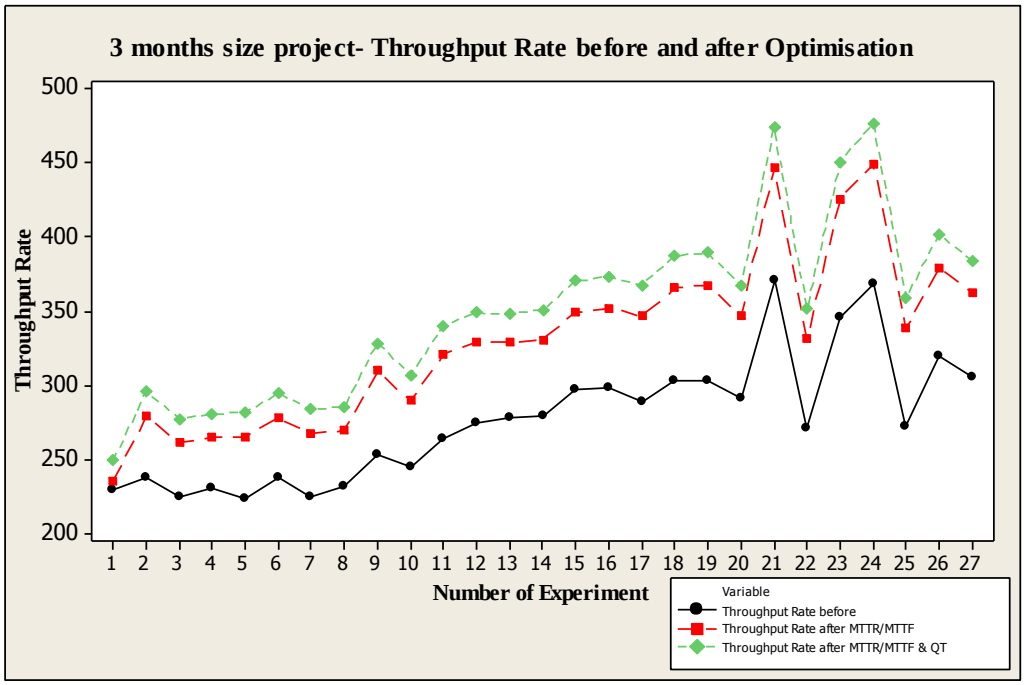

Figure 5.17: 3Months Size Project Throughput Rate Before and after Optimisation

Throughput Rate is being increased as well after applying optimisation.

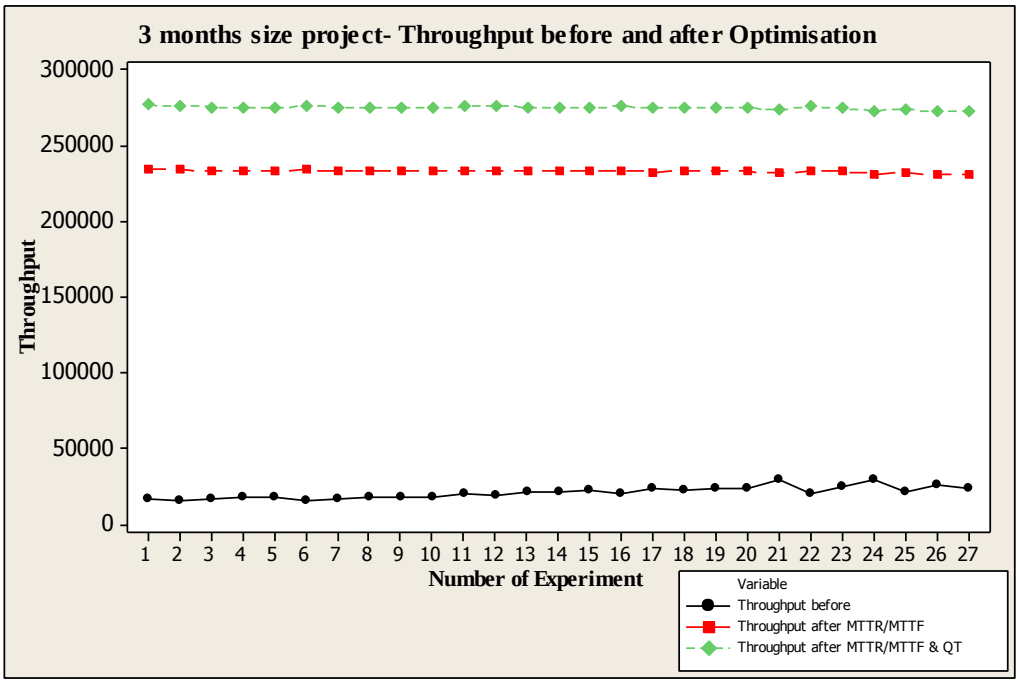

Figure 5.18: 3Months Size Project Throughput Before and after Optimisation

In addition, Throughput is increased as shown in Figure 5.18 while lead time is reduced as shown in Figure 5.19.

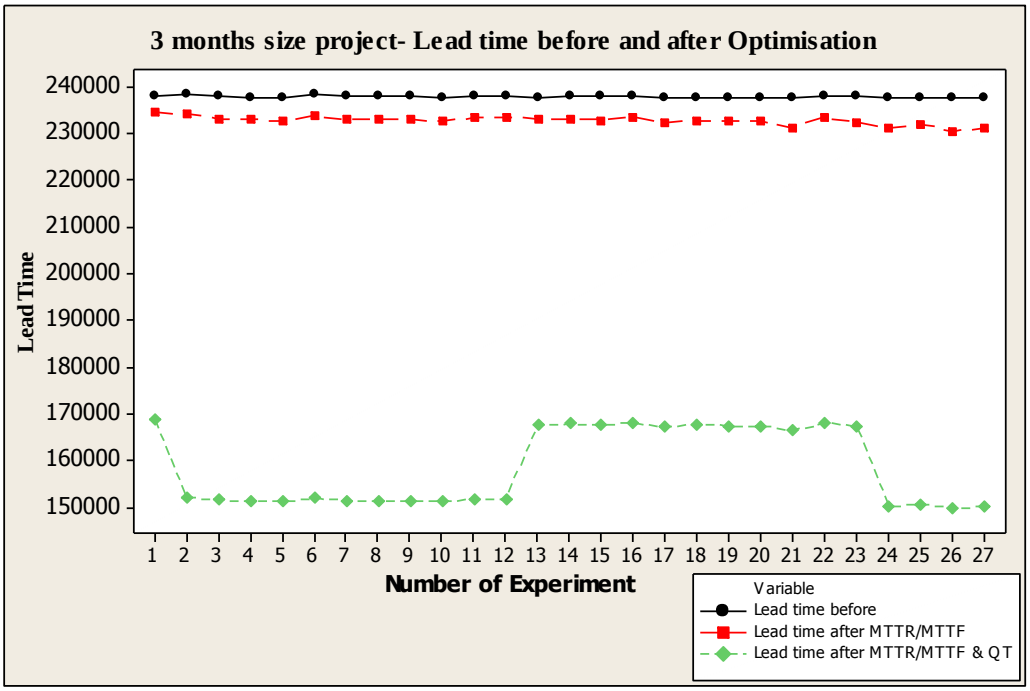

Figure 5.19: 3Months Size Project Lead time before and after Optimisation

In view of that, improvements are been achieved in project completion time and its throughput as shown in Table 5.12 below:

%Reduction in Project Completion Time after MTTR/MTTF Opt.	2.1%
%Reduction in Project Completion Time after MTTR/MTTF & QT Opt.	31.9%
%Increase in Throughput after MTTR/MTTF Opt.	91%
%Increase in Throughput after MTTR/MTTF & QT Opt.	15.2%

Table 5.12: Improvements in 3months size project

II. 6months Size Project:

The following Tables 5.13-5.14 present the results of performance measurements after optimising the MTTR/MTTF and QT for 6months project i.e. 57600 minutes.

a. After MTTR/MTTF Optimisation:

Average %Waiting	Average %Blocking	Average %Stoppages	Average %Working	Throughput Rate (Modules/Minute)	Lead Time (Minutes)	Throughput (Modules)
4.16	16.92	12.69	66.24	236.68	212029.89	28188.61
6.06	17.64	16.07	60.22	245.45	211558.55	30308.18
3.09	14.66	15.36	66.89	230.83	211514.09	31711.50
2.89	15.06	12.54	69.52	237.39	211389.58	32713.88
1.67	15.78	9.28	73.26	229.88	211940.96	31446.09
5.63	17.18	14.22	62.98	244.29	211558.55	30946.05
2.08	14.54	16.79	66.59	231.60	227325.07	34295.40
1.95	14.97	18.42	64.66	238.14	211625.25	31930.20
5.38	19.91	8.65	66.07	261.09	258348.74	39582.68
4.07	15.83	23.22	56.88	251.92	211522.09	37243.70
3.67	16.90	8.17	71.26	271.55	227262.99	38142.87
5.94	19.58	8.81	65.67	282.49	258473.74	46967.95
3.67	15.79	14.78	65.76	285.88	211638.59	39430.70
3.16	18.01	5.63	73.21	287.34	242931.19	48128.55
2.55	15.10	15.16	67.20	306.63	211585.23	35971.59
5.22	17.65	20.00	57.14	306.96	242660.61	50001.33
2.33	16.04	5.78	75.85	296.92	211514.09	47354.32
4.69	18.62	6.84	69.85	312.47	211474.96	42536.24
2.76	15.90	10.89	70.45	312.37	227025.14	45687.07
1.82	14.02	12.98	71.17	300.00	211425.15	52919.70
1.55	17.69	7.85	72.91	381.60	258604.17	44714.84
3.34	17.05	20.59	59.03	279.52	211528.31	44924.63
4.37	17.07	15.95	62.61	355.86	242726.98	62747.54
3.24	16.33	8.15	72.28	379.49	211505.19	38316.24
3.07	16.50	10.49	69.94	280.45	211496.30	47202.75
2.48	15.82	5.80	75.90	329.63	211438.49	43720.86
3.49	17.25	5.33	73.93	314.37	624607.20	23989.50

Table 5.13: 6 months Project after MTTR/MTTF Optimisation

b. After MTTR/MTTF and Queuing Time Optimisation:

Average %Waiting	Average %Blocking	Average %Stoppages	Average %Working	Throughput Rate (Modules/Minute)	Lead Time (Minutes)	Throughput (Modules)
3.52	16.09	9.04	71.35	241.62	190826.90	38054.62
4.59	16.71	12.42	66.27	250.58	190402.70	40916.04
2.55	14.47	11.71	71.27	235.64	190362.68	42810.53
2.17	14.94	8.89	74.00	242.34	190250.62	44163.73
0.89	15.29	5.63	78.18	234.68	190746.86	42452.22
2.73	16.52	10.57	70.19	249.39	190402.70	41777.17
1.81	14.33	13.14	70.72	236.44	204592.57	46298.79
1.47	14.97	14.77	68.78	243.12	190462.72	43105.77
4.47	18.30	5.00	72.24	266.54	232513.87	53436.61
3.11	15.41	19.57	61.90	257.17	190369.88	50278.99
2.09	16.15	4.52	77.25	277.22	204536.69	51492.87
2.88	17.95	5.16	74.01	288.39	232626.36	63406.73
2.75	15.24	11.13	70.88	291.84	190474.73	53231.44
2.68	17.21	1.98	78.13	293.34	218638.07	64973.54
2.10	15.03	11.51	71.36	313.03	190426.71	48561.65
2.72	16.38	17.35	63.56	313.37	218394.55	67501.80
0.96	15.17	2.13	81.74	303.12	190362.68	63928.33
3.37	16.64	3.19	76.79	318.99	190327.46	57423.92
2.24	15.22	7.24	75.30	318.89	204322.63	61677.55
1.29	14.11	9.33	75.27	306.27	190282.64	71441.60
0.72	16.24	4.20	78.85	389.57	232743.75	60365.03
1.48	16.34	16.94	65.24	285.36	190375.48	60648.24
3.31	16.35	12.30	68.04	363.29	218454.28	84709.17
2.59	15.54	4.50	77.36	387.41	190354.67	51726.92
2.39	16.13	6.84	74.64	286.30	190346.67	63723.71
1.60	15.68	2.15	80.57	336.51	190294.64	59023.17
1.89	15.89	3.68	78.54	320.93	562146.48	32385.83

Table 5.14: 6 months Project after MTTR/MTTF and Queuing Time Optimisation

Figures 5.20-5.26 present plots that compare the results for each of the identified performance measurements before Optimisation and after MTTR/MTTF and Queuing Time optimisation.

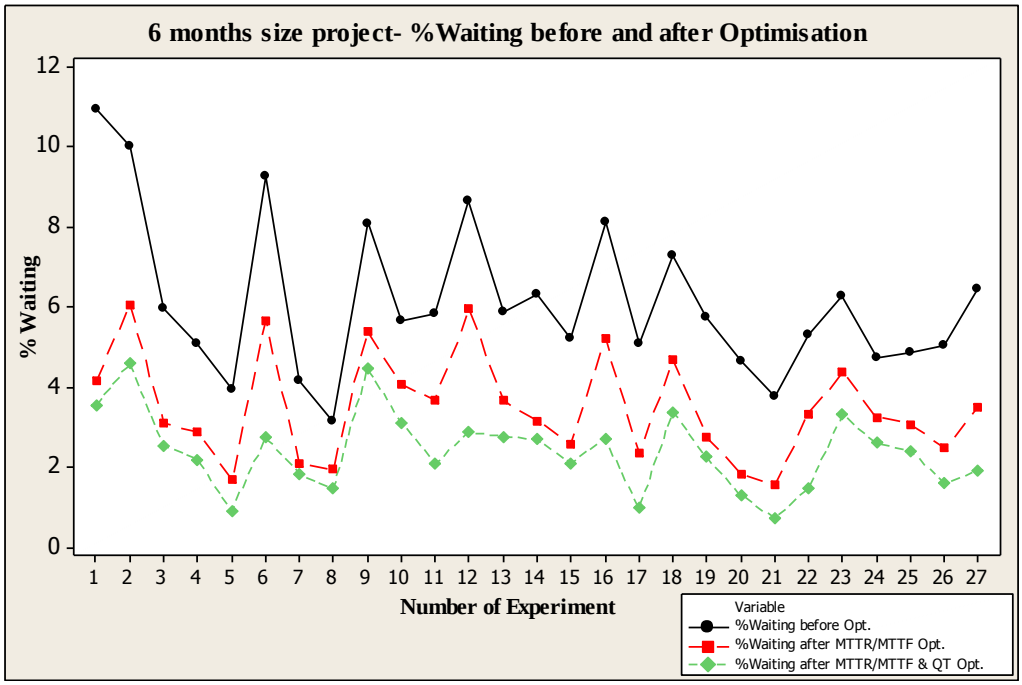

Figure 5.20: 6Months Size Project %Waiting Before and after Optimisation

Previous figure shows the reduction in the %waiting, e.g. it is reduced from 9% to 3% in experiment 6.

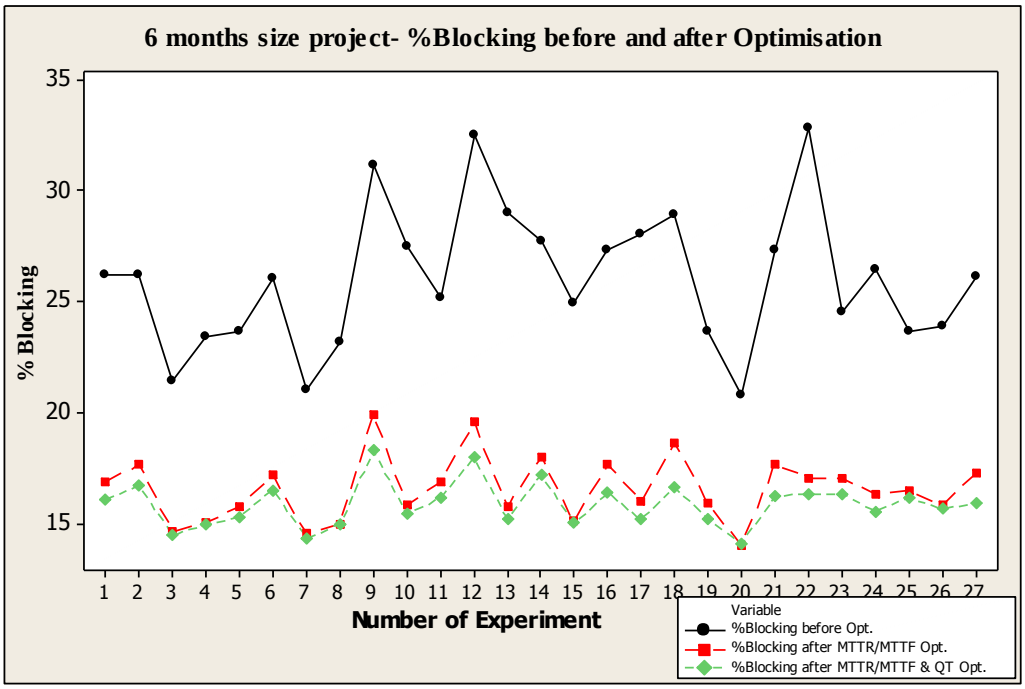

Figure 5.21: 6Months Size Project %Blocking Before and after Optimisation

%blocking is reduced as well after MTTR/MTTF and Queuing time optimisation as shown in Figure 5.21. This reduction can enhance %utilisation.

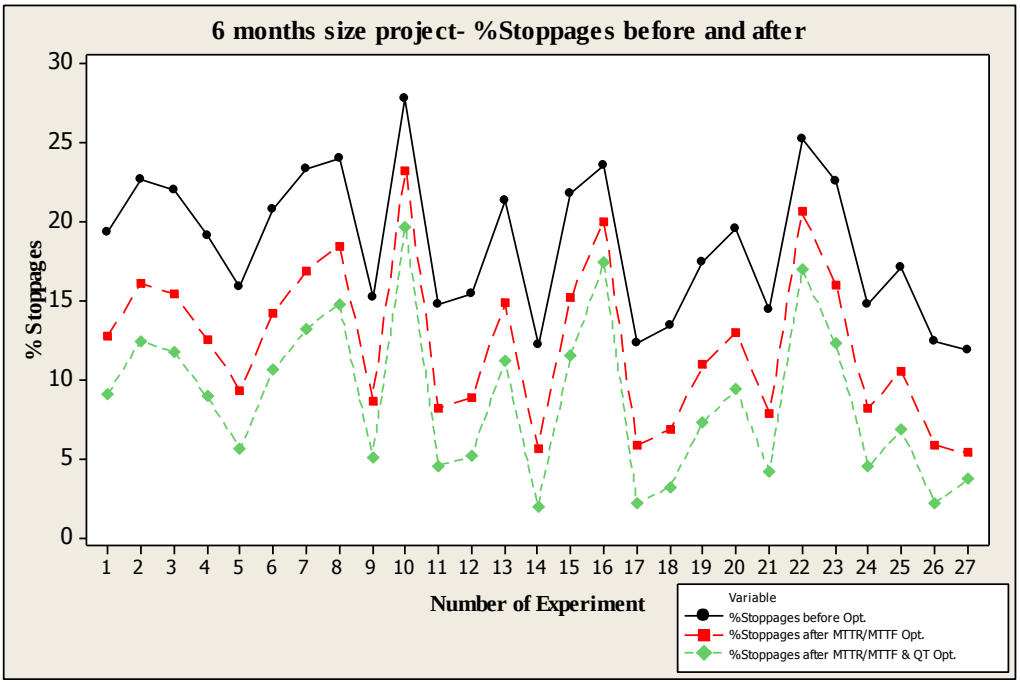

Figure 5.22: 6Months Size Project %Stoppages Before and after Optimisation

%Stoppages is enhanced as well after applying optimisation. For example, experiment#10 shows reduction from 16% to 4%.

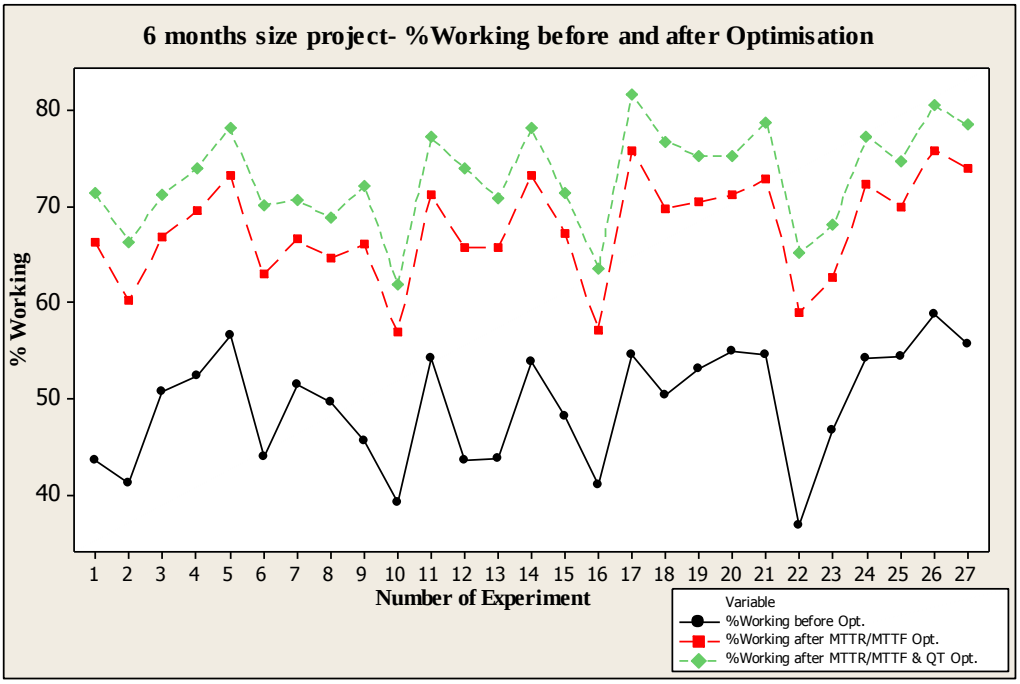

Figure 5.23: 6Months Size Project %Working Before and after Optimisation

The optimisation improved the %working as shown in Figure 5.23. Therefore, reduction in previous performance measurements improved %working.

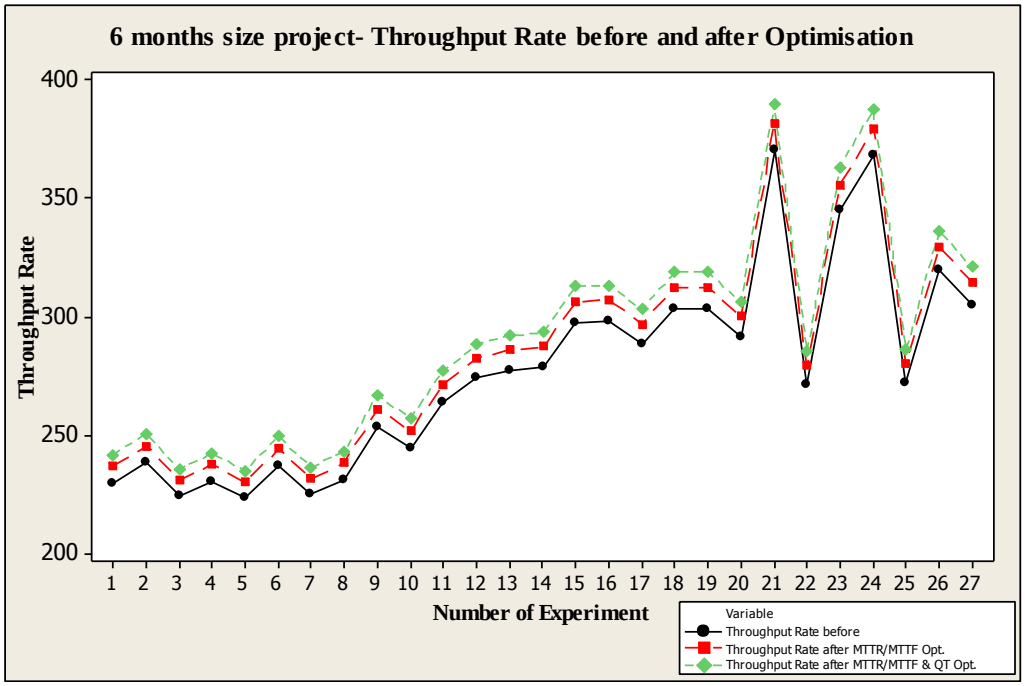

Figure 5.24: 6Months Size Project Throughput Rate Before and after Optimisation

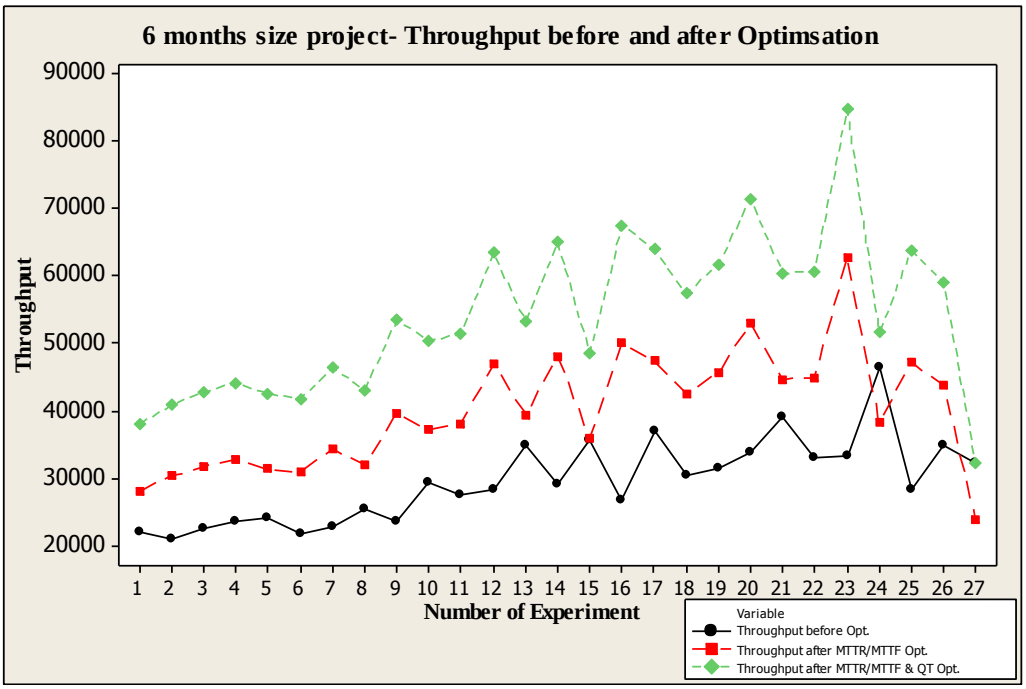

Figure 5.25: 6Months Size Project Throughput Before and after Optimisation

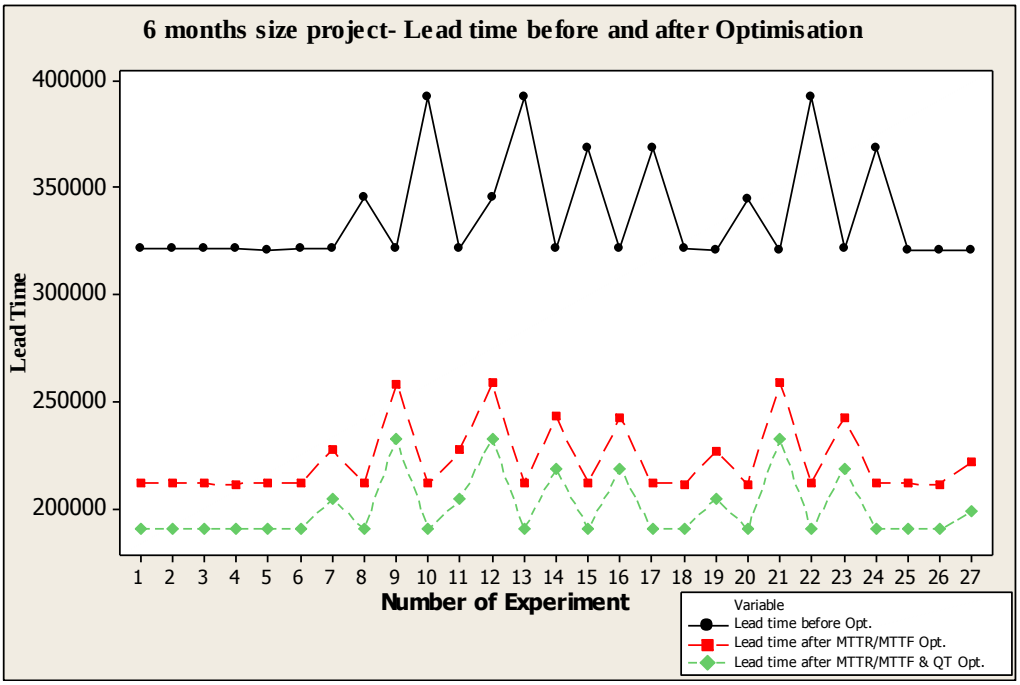

Figure 5.26: 6Months Size Project Lead time before and after Optimisation

Analysis to the previous results coined the improvement that introduced through optimising MTTR/MTTF and Queuing time. Table 5.15 shows such improvement percentages:

%Reduction in Project Completion Time after MTTR/MTTF Opt.	28%
%Reduction in Project Completion Time after MTTR/MTTF & QT Opt.	10%
%Increase in Throughput Rate after MTTR/MTTF Opt.	24%
%Increase in Throughput Rate after MTTR/MTTF & QT Opt.	25.9%

Table 5.15: Improvements in 6months size project

III. **9months Size Project:**

The following Tables 5.16 presents the results of performance measurements after optimising the MTTR/MTTF for 9months project i.e. 86400 minutes. As there were improvements in previous types, the research applied only MTTR/MTTF optimisation to 9months size project.

a. After MTTR/MTTF Optimisation:

Average %Waiting	Average %Blocking	Average %Stoppages	Average %Working	Throughput Rate (Modules/Minute)	Lead Time (Minutes)	Throughput (Modules)
3.03	15.95	6.50	89.06	241.28	262298.04	25267.7
3.14	16.29	6.22	88.19	250.22	262738.84	23548.5
2.04	14.52	6.71	91.74	235.31	262154.78	25319.1
1.52	14.89	4.17	94.06	242.00	280175.52	28318.5
0.45	14.74	1.76	97.15	234.35	261945.40	27068.6
0.71	15.09	1.21	96.67	249.03	262628.64	24308.9
1.33	14.29	8.47	91.32	236.10	262154.78	25605.8
1.18	14.74	5.08	93.77	242.77	262264.98	26420.1
3.54	16.75	4.86	87.31	266.16	262237.43	26420.1
2.16	15.20	5.26	92.27	256.81	299863.40	31858.1
0.45	14.71	1.20	97.39	276.83	262109.60	32001.9
0.31	14.72	0.95	97.40	287.97	262193.35	30514.2
1.42	14.73	3.86	93.72	291.43	262055.60	33019.8
1.89	16.51	3.65	91.69	292.92	262253.96	33881.1
1.11	15.06	4.11	94.19	312.59	244105.92	33224.2
0.80	15.06	1.62	95.88	312.92	262187.84	30622.7
0.19	14.54	1.05	98.04	302.69	261895.81	37073.8
2.11	15.64	3.06	92.64	318.54	262099.68	35111.2
1.74	14.97	4.47	93.04	318.43	280123.69	38708.6
0.90	14.16	3.43	95.85	305.83	261918.95	36211.2
0.23	14.98	1.20	97.48	389.01	261989.48	45050.7
0.37	14.82	1.61	97.02	284.95	262187.84	31144.8
2.17	15.74	4.01	91.93	362.77	262117.31	38244.4
2.09	15.24	3.84	92.58	386.86	261967.44	46524.6
1.59	15.77	4.15	92.71	285.90	268414.94	33718.3
0.66	14.93	1.36	96.94	336.03	262077.64	40559.4
0.66	14.85	0.77	97.77	320.48	271040.70	38862.9

Table 5.16: 9month Project after MTTR/MTTF Optimisation

Figures 5.27-5.33 present plots that compare the results for each of the identified performance measurements before Optimisation and after MTTR/MTTF optimisation.

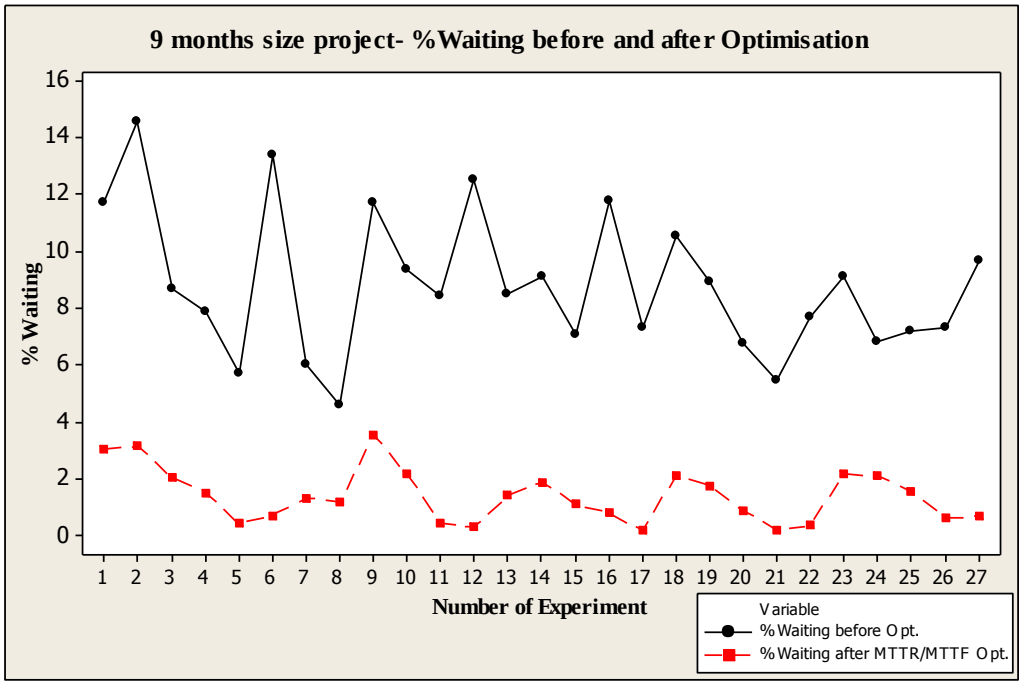

Figure 5.27: 9Months Size Project %Waiting Before and after Optimisation

%waiting is reduced according to Figure 5.27 because of MTTR/MTTF optimisation.

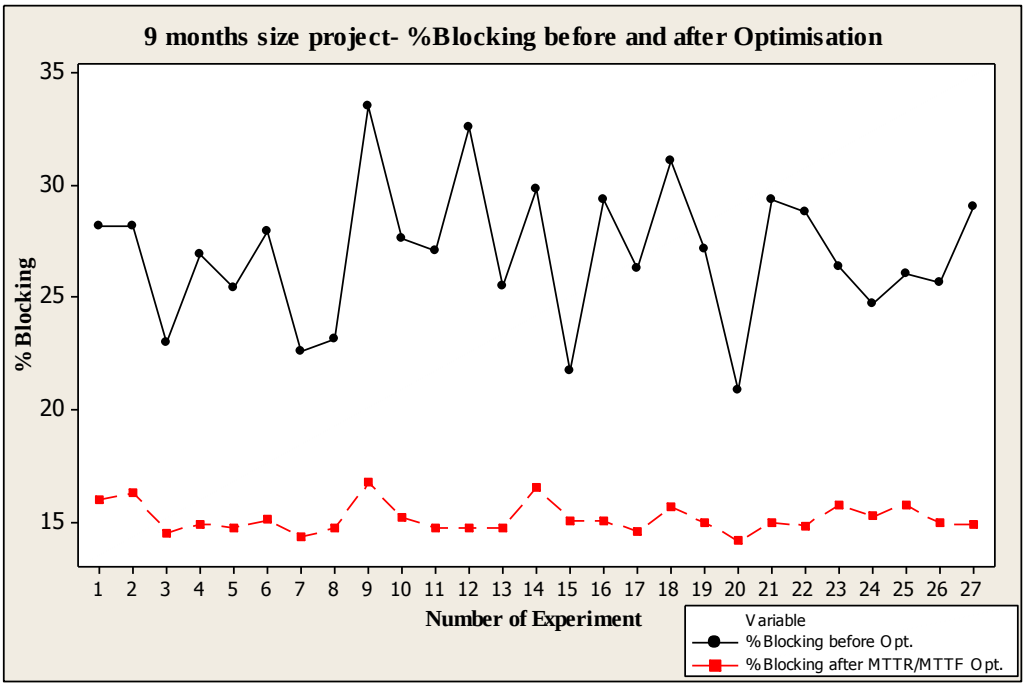

Figure 5.28: 9Months Size Project %Blocking Before and after Optimisation

%blocking is reduced as well with 9 months size project after optimisation as shown in Figure 5.28.

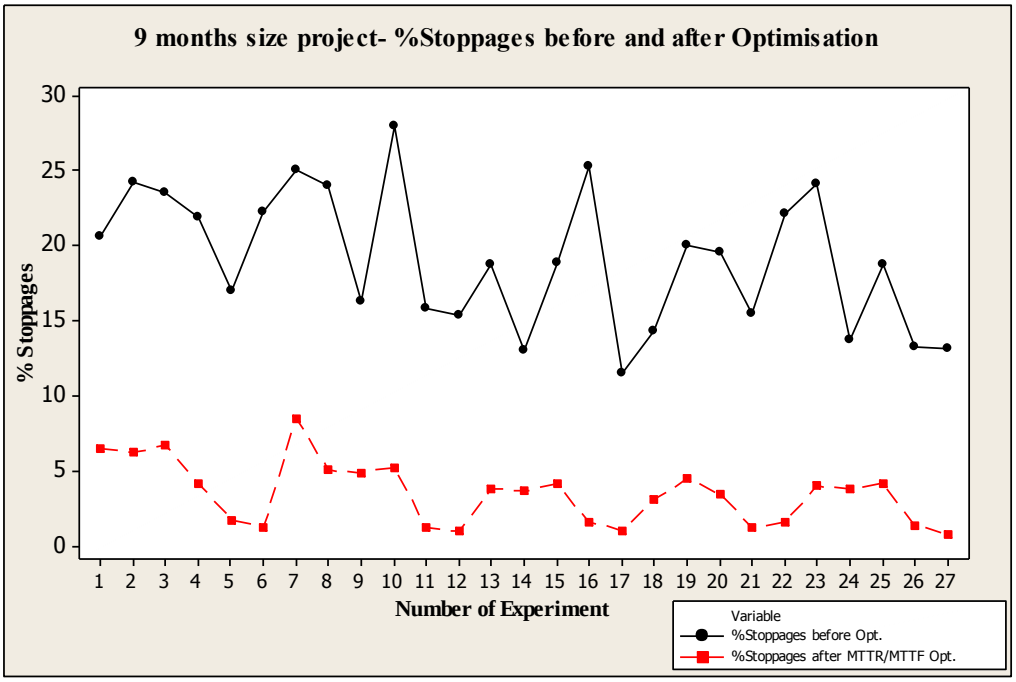

Figure 5.29: 9Months Size Project %Stoppages Before and after Optimisation

%Stoppages is reduced after MTTR/MTTF optimisation, e.g. it is reduced from 13% to 4% in Experiment#17.

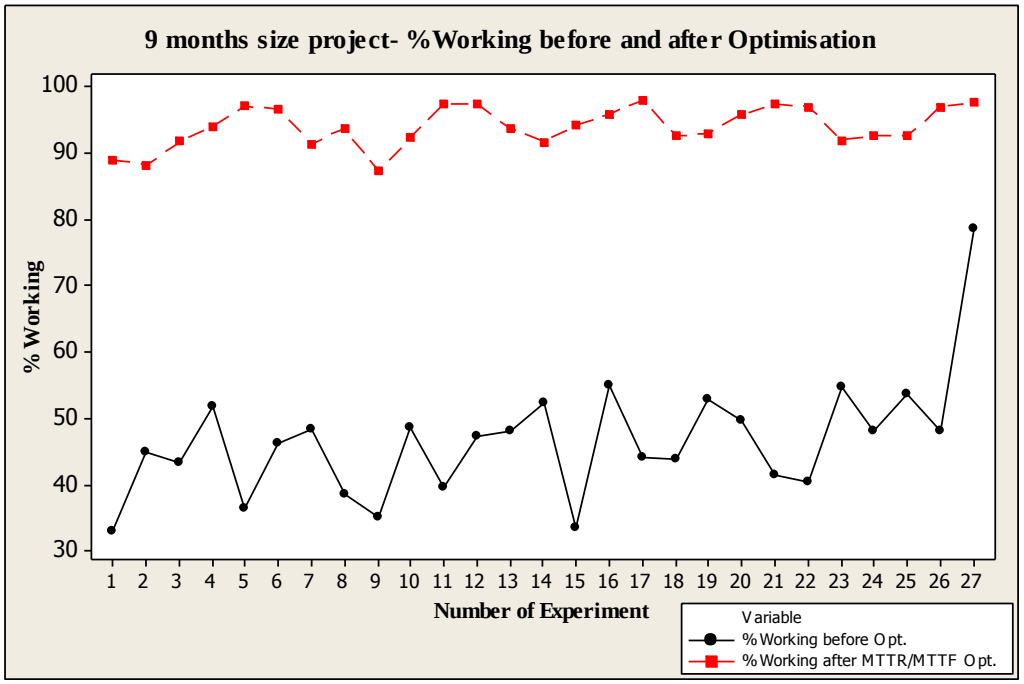

Figure 5.30: 9Months Size Project %Working Before and after Optimisation

According to previous graphs, reductions in %waiting, %blocking, and %stoppages is been shown. This helped to improve the %working to be increased, e.g. experiment#9 showed improvements in %working from 35% to 88%.

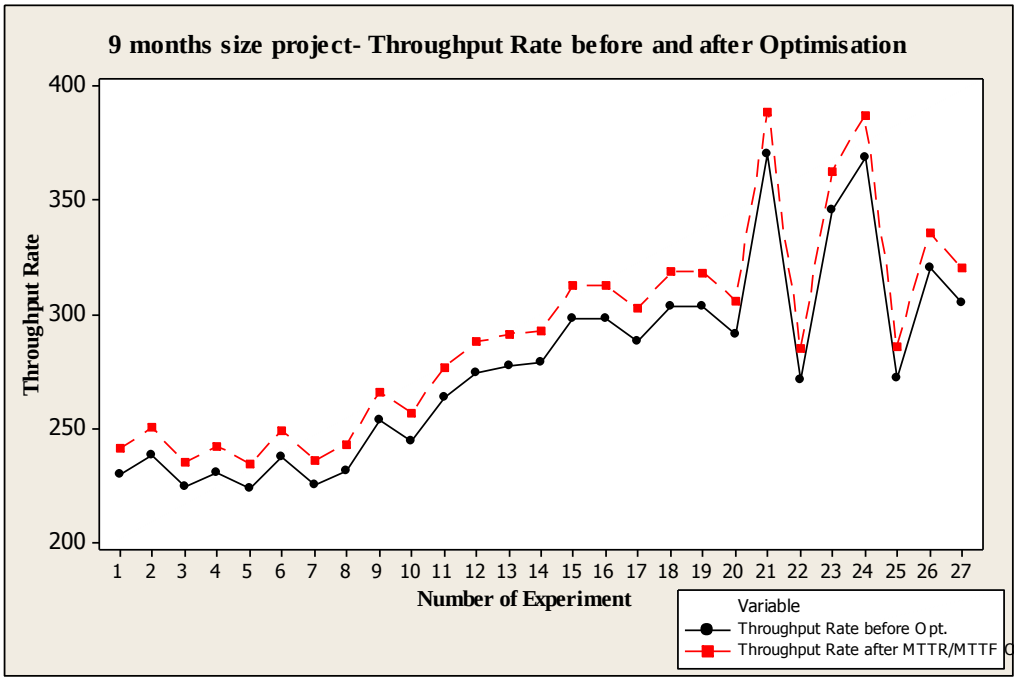

Figure 5.31: 9Month Size Project Throughput Rate Before and after Optimisation

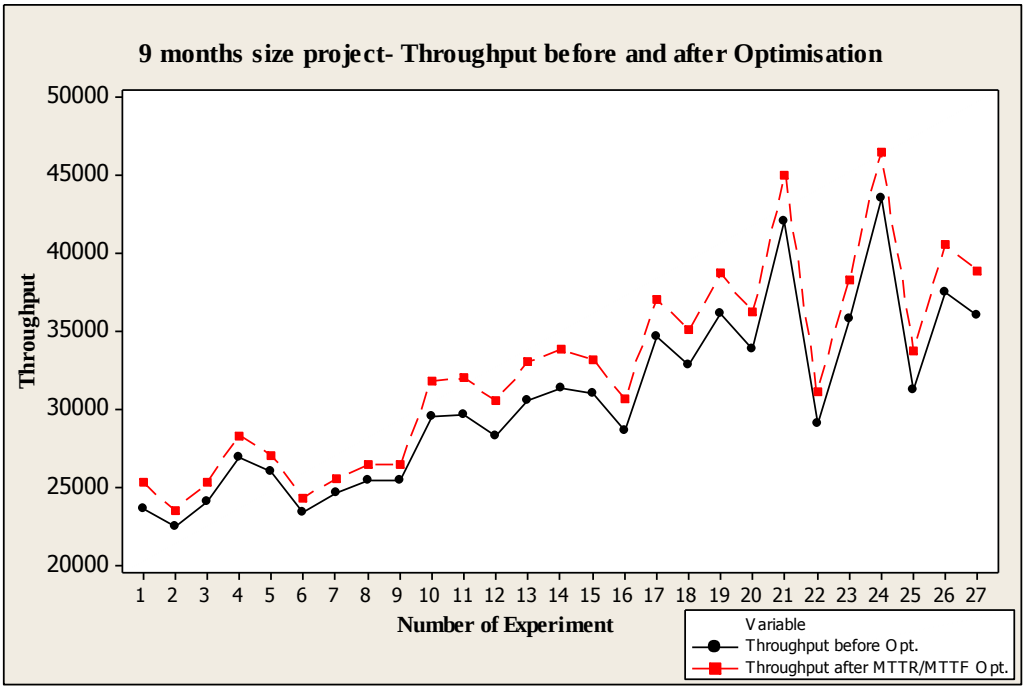

Figure 5.32: 9Month Size Project Throughput Before and after Optimisation

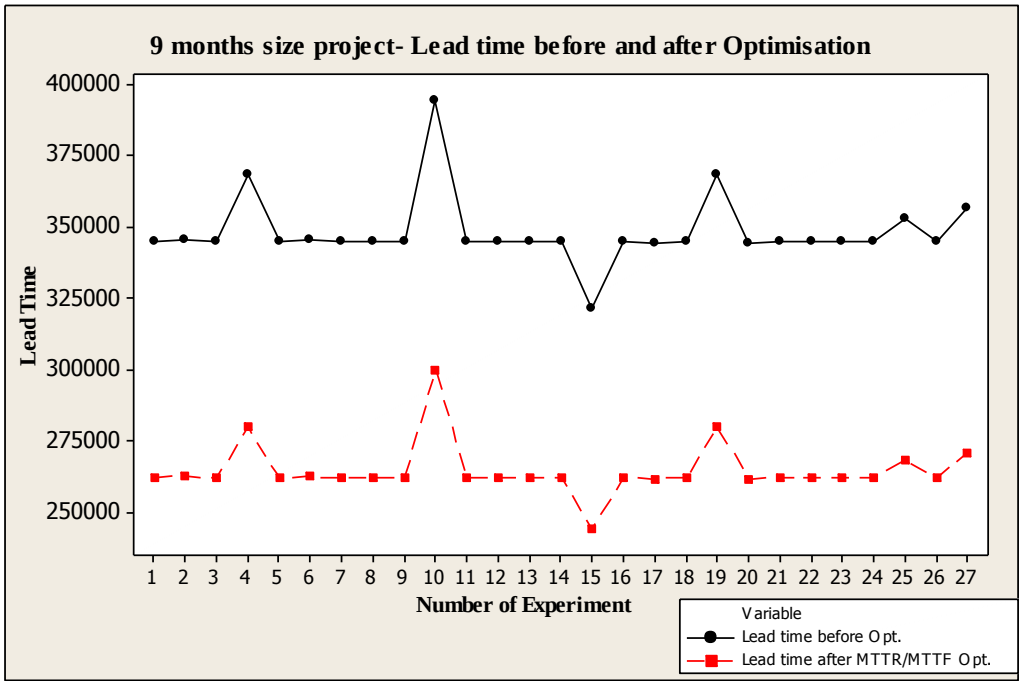

Figure 5.33: 9Months Size Project Lead time before and after Optimisation

Moreover, Table 5.17 illustrates the percentage of improvement after MTTR/MTTF optimisation to the 9 months size project.

%Reduction in Project Completion Time after MTTR/MTTF Opt.	24%
%Increase in Throughput Rate after MTTR/MTTF Opt.	6.1%

Table 5.17: Improvements in 9months size project

Therefore, a recall to Figures 5.13-5.33 and Tables 5.10-5.14, illustrated some improvements to project completion time and throughput rate. These improvements are accomplished through the focus on optimising the outcome of Regression analysis where the ten software factors are reduced into two critical software factors. Therefore, the recommended optimum solution can be easily implemented to enhance these factors with most pay back value to achieve customer demands.

Chapter 6: Discussions

6.1 Introduction

Through empirical examples, improving the plan of project increases its likelihood of success (Zwikael, 2009). Project planning in most cases answers the different questions that can raise during execution period i.e. what can be done, when, where, and what resources to be used in order to achieve project deliverables. According to Kerzner (2006), producing a good plan helps in clarifying project objectives, reducing uncertainty, enhancing tasks and operations efficiency. Moreover, developing a planning model needs to consider the limitations of resources, fuzzy durations, crashing costs, and risks events (Weglarz et al. 2011). The research here has proposed a model that can help in previous points and promotes optimum solutions to achieve customer demands.

Lean had provided the research here with an effective way to study and identify the variables through the different project planning activities. In addition, the structure of the developed model is based on a lean tool, QFD. OPMD structure consists of a matrix view that captures the customer demands and combines them with the identified software generic variables, performance measurements, and risk factors as shown in Figure 5.6.

Results from previous chapter showed that the proposed Operations Project Management Deployment (OPMD) model may have a positive impact on managing and planning the different project phases. In the case when there is no match between customer demands and performance indicators, the optimisation will acts as an iterative

manner to manage the critical constraints to meet customer demands where improvements are being recorded.

The automation feature on OPMD enhances the flexibility needed by management team in order for a faster react to changes by making choice between alternative actions in the situation of uncertainty in earlier planning stages and throughout the project lifecycle. This chapter clarifies the use of best practices from existing software models and compares the developed method with existing ones. Moreover, this chapter discusses the outcome results from the designed experimentation and major findings from previous chapters. In addition, it explains the use of the proposed method in the management field.

6.2 Adopting Best Practices from Existing Software Development Models

Studying existing software models in Section 3.5 provided a comprehensive knowledge of the different practices introduced by them to tackle project activities. Through the development of the proposed model 'OPMD', best practices have been adopted based on their support to planning activities and alignment with Toyota 14 ways as follows:

a. Mapping down the process

The research has mapped the software project based on the stages of the waterfall model that is presented in Section 3.5.1. This practice is the first step in improving any process as it helps in visualising and grouping the different tasks and constraints among the project phases.

b. Adopting risk assessment

Risk assessment has been added to OPMD matrix through the 82 generic risks checklist identified in Table 5.3. This practice is been seen in the spiral model in Section 3.5.2 at the end of each iteration. It is important to visualise the possibilities of risk factors not only in the planning stage ahead of executing but also during execution of the plan. Preparing for risk factors help to save the project from failure and reduce the percentage of cost after their occurrence.

c. Adopting short feedback loops with both customer and employees

Through delivery of prototypes, the design gets validated and ensures fast feedback from customer as presented in evolutionary prototype, spiral, LSD, and staged delivery models in Sections 3.5 and 3.6. Opening discussion meetings with teams' members increase communication, manage constraints, and tasks progress. This practice adopted in OPMD by allocating two stages in the mapped process, i.e. design phase and prototype phase, for a short loop to customer.

d. Focus on customer demands to success

The plan is steered according to customer demands. They are captured and used as the goals or targets of the project, not to be manipulated through the project lifecycle. In OPMD, customer demands are the targets in which the other planning activities are tailored to meet. The optimum solution will be triggered as managing and meeting completion date and throughput rate requested by customers. Evolutionary prototyping and LSD models in Sections 3.5.5 and 3.6 had used this type of practice that focus on customer requirements.

e. Involve team members in plan activities:

According to Lewis (2001), one-sided planning is a common mistake that can affect the success of a project. Involvement of team members in planning activities provides more accurate estimates of the needed period for a project which illustrated in LSD model (Section 3.6). The OPMD has adopted the practice of involving the team members by specifying a phase in the mapped process for discussion as shown in Section 4.4 Step 2. In addition, this practice is recommended by Toyota 14 ways in illustrating that no barriers shall exists from understanding the work and empowering team members.

6.3 Differences between OPMD and Previous Planning Models including QFD

The research proposed method is generic in the sense that it can suite any size project as it has been tested in Section 5.2 Step 7 on three different sizes in addition to its applicability to different sectors than services as will be shown in Section 6.6.7. It proposes steps that allow the management team to plan and control project operations as presented in Section 6.9. The structure, tools, and techniques applied in the OPMD are different than the original QFD.

What differentiates this model from mentioned plan and control models and QFD is been shown in Table 6.1;

Quality Function Deployment (QFD)	Operations Project Management Deployment (OPMD)
• Statically captures the customer requirements according to their needs in a simple language.	• Captures the customer requirements with attention to operational perspective i.e. lead time, cost, throughput, and etc.
• No automation exists. Changing initial customer requirements is effort consuming even with middle size metrics.	• The OPMD is integrated with a real life project simulation model. Any changes to be made during the phases of the project will update the different measures i.e. output of simulation runs automatically to highlight if there is any problem in meeting customer demand.

• No investigation for the different levels of variability that can exist in each project.	• Analysis to the variables/factors affecting the planning process and execution of project e.g. 10 software factors in Section 3.9.
• Risk is considered through competitive technical benchmarking and analysis to market.	• 82 Risk factors grouped down into five groups, which are updated according to the different simulation run measures output through the use of checklist as shown in Section 4.3.4 to indicate any risks involve in meeting any deliverables or customer demand.
• Plan is built according to requirements' weights and their priority orders. Their weights are based on their relative important to customer satisfaction, market competition, and the alignment of new service to customer strategic goals.	• Taguchi OA is used to help modelling the various combinations of the software factors i.e. the different "if" that can occur in different project phases.
• No optimum solution.	• Regression is used to measure the factor that can affect different simulation runs performance measures. Hence, optimisation is used as a response tool for uncertainty or bottleneck to manage/plan different resources and capacity management as illustrated in Section 5.2 step 7 i.e. not only help in managing different resources but can control the complete of the activities among the different project phases.
• Mapping the assessment, interrelationships and interdependencies between the technical responses.	• The OPMD has used regression to identify the output of the running the simulation with the identified 10 software factors which in this case %Waiting, %Blocking, %Stoppages and %Working with project completion outputs i.e. throughput, throughput rate and lead time.

Table 6.1: Differences between QFD and OPMD

6.4 Discussing the Results

According to literature reviews in Section 3.4, existing software models are been studied to give a clear picture of every model advantages and limitations. In which their best practices are examined and adopted in the proposed method as presented in Section 6.2.

- The proposed method started by mapping the process that did not only help in understanding project phases but helped in selecting factors that might control the completion of every identified phase activities.
- Meetings took place with the IT team from ADNOC Company which provided the needed data to build the simulation model and to identify the different types of interruptions during the different phases of the project. In addition, experts from the company had validated the research proposed steps and the levelling of the Taguchi Orthogonal Arrays.
- Accordingly, generic factors were chosen that can be changed according to different type of project/team bases process improvements whether it is a service or manufacturing ones which will enable project team to manage and plan resources and constraints.
- By adopting lean thinking, the factors were chosen to include customer and management involvement through different stages of the project and their input will help in improving projects measures.
- The research undertook different experiments which represented different scenarios that could occur in real project life cycle i.e. different 'if' scenarios which would give a clear picture of how to manage and to react when any risk occur as shown in Table 5.3. Orthogonal Array was selected as a structured balanced method that could identify different scenarios with different levels as illustrated in Table 4.7.
- By identifying different scenarios would give a good standing of each operational factor that can affect the completion of tasks in every phase as in Table 4.8.

- Standard performance measures identified in Bhasin (2008) were the factors chosen in the proposed model, they are:
    - Project Throughput, i.e. it is taken in this research to be the completed modules that are delivered to the end customer.
    - Project Lead time, i.e. it is the project duration requested by the customer.
    - Project throughput Rate, i.e. it is the completed modules per unit time.
- By identifying %Waiting, %Blocking, %Stoppages would not only allow project management team examine the different constraints but also helps examining the relationship between non-utilisation and different performance measures. Accordingly, regression was used, as illustrated in Section 5.2 Step 8, in which showed that %Waiting had a strong relationship to throughput, throughput rate and lead time.
- Taguchi analysis was used to investigate which factors had a great influence on %Waiting as shown in Figures 5.10-5.12 which include;
    - Requirements volatility i.e. modelled as Mean-time-to-Repair (MTTR)/Mean-time-to-Failure (MTTF), and
    - Communications Overhead i.e. modelled as Queuing Time.
- However, by looking at Taguchi analysis results, size of team seemed to have a great effect as well on %Waiting. However, by minimising the effect of waiting time will help in planning where and when resources are needed i.e. understating the level of variability or the constraints that could occur in every activity will help in managing different resources along the project phases.

- By using evolutionary optimization algorithms, the research integrated optimisation with the different scenarios to identify the optimal schedule for MTTR/MTTF, Queuing time, and estimated the resources required to complete each activity in the project phases with minimum. With the support of the model, time variation could be determined and reduced, so as to break the bottleneck of managing project changes in practices.
- The optimisation helped in improving MTTR/MTTF and Queuing Time in 3 months and 6 months size project. The significant improvements outcome in project completion time and its throughput in previous sizes suggested to only optimising MTTR/MTTF in the case of 9 months size project which also showed positive improvement in its performance.

  And so, the following are main improvements outcome for each of the performance measurement adopted in the model:

  - The proposed model had improved the project throughput in relation with each size project as presented in Table 6.2:

Size of project	**Before applying OPMD**	**After applying OPMD**
3 months	min 15467- max 29987	min 272492- max 276886
6 months	min 20880- max 46480	min 38055- max 84709
9 months	min 22427- max 43480	min 23548- max 46524

Table 6.2: Throughput Improvements

  - Lead time has been reduced after the use of the proposed method as shown in Table 6.3:

Size of project	**Before applying OPMD**	**After applying OPMD**
3 months	min 237655- max 238420	min 149744- max 168735
6 months	min 320895- max 392568	min 190251- max 562146
9 months	min 321192- max 394557	min 244105- max 299863

Table 6.3: Lead Time Improvements

- Throughput rate has been improved significantly after the application of the proposed model. Table 6.4 below illustrates such improvement according to each size with minimum and maximum values:

Size of project	Before applying OPMD	After applying OPMD
3 months	min 223- max 370	min 249- max 476
6 months	min 223- max 370	min 235- max 390
9 months	min 223- max 370	min 334- max 389

Table 6.4: Throughput Rate Improvements

Previous observations and findings had validated the use of the developed method and ensured whether an improvement can be made with this method in planning and managing the software project. The method helps to plan the project through examining the different scenarios and suggesting the optimum values for the most critical factors. Therefore, adjusting the levels of the identified software factors/variables/parameters and controlling their variations may result in a low-cost solution.

6.5 Applying OPMD with Existing Planning Models

As mentioned earlier, the developed method can be used as a quick responsiveness tool that will visualise the different if scenarios and how the team member can react to any uncertainty or if risks occur that can affect project deliverables or milestones. In other words, it can identify bottleneck and the different constraints and move resources accordingly when and where they are needed.

Although there are lot of project management techniques dealing with software development projects, planning and carrying out projects tasks can be tricky for a variety of reasons, making the ability to complete them successfully is a very valuable asset to any employer.

According to the empirical analysis accomplished by Zwikael (2009), time factor played the greatest effect on project success and competition. This explains the vast effort put into controlling and time management throughout literatures. Different techniques had been developed include project evaluation and review technique (PERT), critical path method (CPM), critical chain analysis (CCA), etc. OPMD can enhance the use of those existing models and vice versa as explained below.

- **Project Evaluation and Review Technique (PERT)** is the first model being used in industry to simplify the plan and control of large size projects (Malcolm et al. 1959). PERT had helped project managers to compute the minimum time needed to complete the project in hand through completing the needed time for each task within the project. A significant point within PERT is the integration of uncertainty (i.e. giving a slack time between different tasks, computing early finishing time and late finishing time). Such incorporation reflects project reality since precise details of project schedule are limited at the start. However, the previous model does not consider resources availability.

- **Critical Chain Project Management (CCPM)** emphasises the effect of limited resources in developing the plan for the project (Goldratt, 1997). This method is an expansion to Goldratt previous method, 'Theory of Constraint (TOC)'. In TOC, Goldratt (1997) had illustrated that any production line/system is limited

by a constraint (e.g. bottleneck) which control its productivity. Therefore, to maximise the system output, a buffer (i.e. feeding buffer) is positioned in front the bottleneck to keep the bottleneck operates continuously in full capacity. Nevertheless, TOC and Critical Chain Project Management had their disadvantage in not considering multi project (Leach 2000; Elton and Roe 1998). In that case, different processes can seek the same resources instantly and this will create a conflict.

The developed OPMD method here aligns with previous existing efforts in aim to manage project planning and overcomes some of the mentioned limitations earlier. It accepts the outcome data of the previous models in attempt of building the simulation model. In addition, OPMD can;

a. Adopt PERT's WBS to map down the process.
b. Use the slack of time in scheduling to input the triangular distribution function as been shown in Table 4.7.
c. Analyse interdependences and critical path to adjust the levels of factors in running Taguchi experiments.
d. Accept the suggestions recommended by previous models to adjust the mapped down process, i.e. allocating a feeding buffer in front of the bottleneck workstation, then rerun the simulation and compare results.

Furthermore, OPMD considers uncertainty through the adoption of triangular distribution function in feeding the simulation model as presented in Table 4.7. Resources in OPMD are considered to be important as two of the software factors identified in Section 3.9 are 'size of team' and 'availability of developers'. Such factors

allow the management to focus on resources available to project. OPMD as will be shown in Section 6.7.3 can be used in managing and planning multiple projects.

6.6 Using OPMD with existing manufacturing Scheduling

Managing tasks in any project are similar to scheduling jobs and resources in a job shop in manufacturing and service sectors. In a job shop, simultaneous tasks with several diverse and low-quantity jobs are using mutual resources to achieve customer orders. This feature puts Job shop in a high-mix, low-volume (HMLV) category (Thiagarajan and Rajendran 2005). The jobs are similar to the projects in having different routings, due dates, priorities, materials and resources requirements. Both pass through a sequence of work centres where %waiting varies in relation with the availability of required resources at those work centres. Moreover, some jobs with numerous tasks dependency relations and governed by finite capacity resources constraints can be considered as projects.

Both Job shop and project tasks complexity arises from the difficulty to measure the total tasks waiting time due to the diverse routings , processing time varies according to job and work centres, continuous change in product mix, and capacity limitation in resources. Therefore, predicting job progress, WIP, lead time, %utilisation, throughput on shop floor is hard and this lead to high uncertainty to implement an accurate plan.

Due to previous factors, planning and control tasks tend to adopt quick, real-time decisions according to experience, intuition, expert views, simple calculations, and etc. In addition, different approaches are adopted to maintain an accurate plan as follow (Thiagarajan and Rajendran 2005, Kim et al. 2003):

a. **Manual Planning/scheduling**, where a scheduler role is used to track job/task progress on the shop floor, collect the needed data, and reports it to management. However, this approach is not effective for complex projects/jobs with simultaneous tasks.

b. **Planning/Scheduling on Whiteboards and Excel Spread sheets**, this is efficient with simple, similar due date tasks where adding electronic version that use Gantt chart with a simple drag-and-drop operations on computer screen extend the use of this time. However, lack of accepting changes in job priorities, fast what-if analysis, and lack of friendly drag-and-drop functionality with friendly graphs representations are some of this type limitation.

c. **Planning by Project Management Software Tools**, which are useful to resemble resource-constrained of multi-project aspects. Still these tools are un-capable to provide neither dependable and fast what-if analysis nor a capacity planning mechanism for production systems. Their main advantage is to represent resource-constrained.

d. **Planning by ERP Systems** that integrats internal and external management information within the organisation to facilitate the flow of information between the two ends, customer/stakeholders, and organisation management team. This type use increased lately due to the use of shop floor data collections systems that can provide instant real-time job status to customers. ERP still lacks the ability to give an intelligent decision support modules where a feasible schedule requires manual repair through drag-and-drop operations in Gantt chart.

Through literatures, other approaches like lean manufacturing, finite capacity scheduling, quick response manufacturing (QRM), and CONWIP are being adopted for production/project plan, control and management. Some of them provide macro details while others provide more details in the project plans.

Previous manufacturing scheduling approaches are indeed good enough for small and simple job shop production systems. In spite of many such advantages, OPMD can assist in overcoming their limitations through;

a. reacting with different job shops and calendar exceptions of resources,
b. changes in job priorities,
c. multiple resource requirements of operations, and fast and extensive what-if analysis, etc. and
d. fast and rapid react to changes to plan during execution.

6.7 Application of OPMD

The proposed method 'OPMD' helps in planning and managing projects. It works as a framework that provides standard steps to enhance the data available and eliminate the non-value-added activities or decisions during the development of project plan. The model can be used in different range of projects as will be shown shortly.

6.7.1 Planning Existing Projects

In the existing projects, the model can help, based on the present available data of the project, to;

1. **Ensure the accuracy of the previously chosen performance measurements**. The use of the performance measurements is to reflect the progress of the project and they need to be aligned with the high strategic level goals. In addition, it is better to have direct relationship between PMs and project goals. Therefore, the model accomplishes that through capturing customer demands first then identifying performance measurements accordingly as presented in Section 4.3.5.
2. **Highlight the major factors affecting the project activities** to be reduced. The model encourages analysis to literature reviews, project characteristics, and data related to the existing project to identify the variability list/factors affecting project operations. This can be accomplished through analysis to the available data for the existing project and its characteristics. This will help in choosing the range for the factors to implement Taguchi OA. For example, if the project is of pure technical type, skill factor is considered to higher effect and will be modelled.
3. **Suggesting the optimum values to the critical factors**. This function is used if the readings of PMs of the existing project are not matching customer demands. Therefore, risk will be high where the use of regression analysis and optimisation can promote values to the identified software factors to satisfy customer demands as shown in Section 4.4.

6.7.2 Planning New Projects

In the case of new projects, the management team can make use of OPMD by going throughout the proposed steps in Section 6.9. They do provide a framework with the ability to suggest the optimal plan in relation with customer demands.

6.7.3 Planning Multiple Projects

The OPMD will apply the same steps mentioned in Section 6.9 to plan each project individually. However, attention will be paid to resources as different projects can seek to use the same resource (e.g. workstation, personnel, tools, equipment, space, transportation, etc.) at the same period (Patanakul and Milosevic 2009; Ruuska et al. 2010). Balance between the available resources is important to ensure smooth and less delay in delivery to end customer (Mota et al. 2009).

In addition to the proposed steps in Section 6.9, OPMD pays attention to the following aspects while planning multiple projects;

1. **Realistic project acceptance**. OPMD had identified 10 generic software factors/variables in Section 3.9 that can affect the planning activities of the project. They can be used as indicators based on their values to decide whether to accept or reject new project while managing others as shown below (Archibald 1975; Ireland 1997);
    a. Appropriateness to project size in having enough resources to match the size of project. OPMD have two factors (i.e. availability of developer and size of team) that can be compared in managing multiple projects as presented in Section 3.9.

b. Schedule, where considering the size of team and available resources can convince or not to accept the project.

The previous points are used in the planning stage when projects can be rejected rather than failing to finish them as promised.

2. **Balancing Resource Allocation**. In OPMD, customer demands provide exact time of delivery; therefore, work breakdown structure will be implemented accordingly. This specifies the major tasks assigned in each project and needed resources.

   Size of team and developers availability are another factors considered by OPMD which help allocating needed size and levelling resources to eliminate the manpower peaks and valleys throughout project lifecycle (Kerzner, 2006). Such analysis helps to have sufficient resources capable to satisfy the needs of multi projects that their timeline cannot be adjusted (through backward or forward) and seek the same particular type of resource at the same time (Greasley, 2006).

   Moreover, one of the risk factors groups is ‘preparation risks’, which raises the issue of resources conflict and highlight the attention to this issue while developing the plan as shown in Section 4.3.4.

3. **Allocating similar tasks in different projects to one team**. Since the different customer demands of each project are captured through OPMD, similar demands can be accomplished by one team if its size is sufficient. Ideal team can be shifted to help lower resources tasks. This is recommended by Toyota 14 ways in Section 2.7 and LSD as shown in 3.6.1.

4. **Balancing trade-off** through the use of customer demands as indicators to decide which project needs more resources. The optimum solution will be given according to the needed completion time and throughput as shown in Section 4.4 Step 7 for that particular project. Therefore, trade-off will be made accordingly.

6.7.4 Planning Project in Different Operational Sectors than Services

OPMD tackles the planning issues shared in any operational sectors (including manufacturing, services, etc). These issues are:

1. It supports the common plan and control steps which can be listed as;
    a. Identification of Goals and Objectives,
    b. Stating work description and Instruction (specification, WBS, Statement of Work SOW).
    c. Master Detailed Schedules.
    d. Time/Cost/Performance Tracking.
    e. Risk assessment.

    Previous steps are seen in developing the simulation model of the project in hand as shown in Section 4.4 Step 4.
2. OPMD takes in consideration two major points:
    a. **Resource constrained**- ensuring the due dates of the different activities of the project with attention not to overload the highly specialised resources as skill is one of the software factors mentioned in Section 3.9.
    b. **Time constrained**- in the need for completion the project in a specific time frame (e.g. due date requested by customer), alternative resources might have to be utilised such as subcontractors. This is implemented in

the model through adding the effect of size of team, their level of skills, and availability of developers while designing Taguchi OA in Section 4.4 Step 4.

According to the previous points, crashing activities (i.e. using additional resources to shorten project completion time) can be considered especially with critical activities (Dodin and Elimam 2008).

Accordingly, OPMD can be applied in different sectors to plan their operations. Only attention is paid to the values of the identified generic ten operational factors/variables for that particular domain/sector which can be modified through adding more factors to suite the project sector environment.

6.8 The Use of the Identified Performance Measurements in OPMD

Analysis to Literature reviews has shown that traditional performance measurement initially focused on financial performance measures e.g. return on investment (ROI), productivity, utilisation, efficiency and profit (Ghalayini and Nobble 1996). Followed that the shift to a non-financial performance measures e.g. lead time, flexible capacity, %waiting, machine utilisation, customer service, etc. especially after the introduction of new technologies and philosophies, such as Computer Integrated Manufacturing (CIM), Just-in-Time (JIT) and Total Quality Management (TQM) (Skinner, 1986).

Specific performance measurements are being identified in the developed OPMD as presented in Section 4.3.3; the emphasis being placed on non-financial measures in

order to gain an overall picture of the project's performance. However, the proposed model is not limited to these PMs; others can be added according to project needs.

The identified PMs in the proposed model provide the following services;

1. **Assuring project on track**. The values of PMs do represent how accurate the project in progressing toward its final goals. If there is lack or biasing in such values, it means that the project is off track.
2. **Identifying the optimal plan**. Based on their readings, the management team can choose which plan suites project's needs. In the case of no match between PMs and customer demands, optimal plan can be chosen according to the suggested values after optimisation to the critical factors with high effect on project performance as shown in Section 5.2 Step 9.

   Moreover, they help to visualise constraints that can occur through different stages of the project and prepare for them, e.g. extra developers in the coding phase, more time in the testing phase, etc.
3. **Highlight Risk Events**. When the project performance measurements readings are not matching customer demands, risk is highlighted. Attention is given to reassess risk by applying the risk checklists given in Table 5.3.
4. They can be used as basic metrics to **experience control** over flow tasks and information through project lifecycle.
5. Establishing a bond between **strategic performance measures** with operational level performance measurements to achieve the organisation strategic goals.

Accordingly, management team can have a clear vision of constraints ahead of time by running different scenarios and collecting data of PMs related to each scenario. Comparison to such data with values of operational factors, the management team can choose the proper scenario to use it as a plan for the project or can prepare the project environment to suite the suggested optimum solutions.

6.9 The Proposed Steps of the Developed Model

The OPMD model proposes the following steps for plan, control, and management of projects. These steps are:

1. Understand the **characteristics of the project environment** through data analysis, data collection, and formal/informal interviews with different types of employees, stakeholders, and customers.
2. **Identify the different operational factors** that affect the plan, control and management of the project. Choosing the operational factors should be based on their direct effect on project operations by either increasing or decreasing certainty, flow of information and materials, quality, customer demands fulfillment, and risks level.
3. Based on step 1, the different activities within each phase of the project should be collected to **map down the lifecycle/stages of the project**.
4. **Develop a simulation model** that imitates the reality of the project according to the data collected through steps 1-3.
5. Define the **customer demands** in a simple, comprehensive, and flexible form. The captured demands should be comprehensive in a sense that contains all

needed facts about project deadline, expected functionality, size, level of quality, etc.

6. **Study the organisation constraints** (Bolton and Dewatripont 1995),
    i. availability of resources,
    ii. budget limits,
    iii. schedule deadline,
7. **Analyse the operational factors levels** in relation with project environment. In that sense,
    i. how skilled the team's individuals,
    ii. how supportive the top management,
    iii. degree of customer involvement,
    iv. level of quality required toward delivered service or product, and
    v. degree of ease in communication between team members and hierarchy of team.

    This analysis should be based on experts point views to identify the range of values each factor has, (maximum, medium, and small). It is named levelling the factors in Taguchi's concept.
8. **Choosing the proper Taguchi Orthogonal Arrays** that match the number of factors and their levels.
9. **Choose the proper key performance indicators** that reflect project progress and provide instant feedback whether project is biasing from its targets or not.
10. **List all expected risk events** and their factors that can cause delay or failure in delivering the promised customer demands. Then cluster them in groups for easy access.

11. **Run the suggested number of experiments/scenarios by Taguchi Orthogonal Arrays**. Data of PMs are collected and compared in relation with the Operational factors.
12. **If the PMs Results match with customer demands, then the chosen values for the generic operational factors is correct** and team management can proceed with this plan.
13. In the case of no matching exists between customer demands and PMs results, the following steps are fulfilled,
    i. Identify which of the Key Performance Indicators has strong effect on project completion time and its throughput using Regression analysis
    ii. Use the DOE to reduce the size of variability list affecting the identified performance measurement in pervious step 13-i.
    iii. **Apply optimisation to identify the optimum solution** for these operational factors in order to achieve customer demands.

Previous steps do provide a guide line to management team to follow in attempt to improve the development of the project plan. However, projects accompanied with high uncertainty and complexity that demands great flexibility and fast reflection to changes during projects execution to success, function, and generate knowledge (Perminova et al. 2008).

6.10 Limitation in Operations Project Management Deployment (OPMD)

The research here did not tackle cost. Cost is a big area that needs more investigation. The OPMD aimed to help in planning and control rather than addressing cost. However,

this model can be improved upon to address cost analysis aspect as will be shown in chapter 8.

Another limitation of the OPMD is the dependability on experts judgement to evaluate the range of different operational factors when comes to levelling them.

Chapter 7: Conclusions

Projects are critical in their high uncertainty levels throughout their phases especially in lack of the visible data about the needed time to finish the required tasks, not meeting customer demands, and the unseen risk factors that can push the project to fail. Having a unified and standard process/model can enhance the planning process activities and reduce effort to achieve project goals.

The research here has mapped down the software project process through a case study, interviews, and literature review in Section 4.4 step 2 where each phase is handled as a discrete or single process. This mapped model is used as a vehicle to test the different levels of variability and their interrelationships that can affect project performance measurements.

The research has underwritten to the body of knowledge the identification of the critical factors affecting software project completion time and its throughput. The research then optimised the critical factors where a positive effect on reducing %waiting, %blocking, %stoppages, and increasing %working has been seen.

The research has succeeded in building an automated planning model entitled ‘Operations Project Management Deployment (OPMD)’, that can help the management team in managing and planning among the different software project phase’s constraints, identify where and when different resources will be needed.

In addition, OPMD provides the following:

a. focus on customer demands to have a successful project as shown in Sections 3.8 and 4.4,
b. early identifications and highlighting of problems and risks events in Sections 3.10 and 4.4,
c. examine the complete alternatives/scenarios the project plan can take in order to identify the effects of variables on project completion time, its throughput and throughput rate,
d. manage the constraints i.e. quick response to uncertainty by using regression which can highlight the factors that can affect completion of project deliverables or milestones (Cho, 2006), and
e. providing the optimum solutions to attain customer demands by using genetic algorithm optimisation as presented in Section 4.4 Step 9. However, optimisation here is acting as quick responsiveness to uncertainty and to reduce the level of risks.

Projects are accompanied with high uncertainty and complexity that demands great flexibility and fast reflection to changes during projects execution to success, function, and generate knowledge (Perminova et al. 2008). Such needs are provided by the proposed automated OPMD model to help in planning and managing them.

The contribution to knowledge from this research includes:

a. Studying and identifying the variables/factors affecting the software project in achieving its promised goals, and integrating these factors with the proposed OPMD model.

b. Analysing the existing software development models in Section 3.4 and adopting the best practices of each model that aligns with the mentioned aims in this research.

c. Proposing a standard steps which can be used as road map for planning and control services operations according to lean philosophy and guidance (Machuca et al. 2007; Martin 2001).

d. Integration of simulation modelling techniques with Taguchi analysis to test the different ‘what if’ scenario.

e. Developing a plan and manage model that provides a clear picture of the project operations activities by capturing the customer demands, breaking the activities into tasks, highlighting the variability types and risk factors may affect project outcome. The model acts as a quick response to uncertainty and when projects outcome are not matching customer demands.

Chapter 8: Recommendations for Future Work

This research has proposed steps that can help in planning and managing the different "if" scenarios or constraints that can affect the completion of different act light on the improvements in planning and managing project activities. Therefore, the following recommendations can be considered for further work:

1. Cost Consideration

The model did not consider cost measurements. Cost is one core element of the Triple constraint triangle (i.e. time, quality, and cost). It plays major role in decision to be made by the management team to either add more resources or accept partial success. Therefore, OPMD can be enhanced further with the calculation of cost in the form of salaries, equipment budget, etc.

2. Managing Human Resources

a. Resource utilisation

The method here is an automated planning and control tool that emphasises not only in fulfilling customer order but will improve the efficiency of the process. OPMD examined different variables that determine employee (people) effort; however understanding of what is needed to meet not only customer demand but organisational objectives needs more investigation. By running the different experiments, the Operations Project Management Deployment (OPMD) can help suggesting where and when employees (operators) may move from downstream to upstream. The research can take the OPMD a step forward by simulating human agent aiming not only in improving

the efficiency of project phases but can act as an empowerment and involvement tool to get the best of people.

b. Change of Culture

As with any introduction of change to improve productivity or quality of project management, the implementation of Lean philosophy faces many obstacles and difficulties (e.g. refuse to accept change by individual or high management, not enough support from key responsible people, and temporary acceptance for the lean practices then abandon them, etc.) (Achanga et al. 2006). Therefore, to overcome previous obstacles and difficulties; organisational culture and management may need further investigation.

c. Adding Workforce Behaviours to Consideration

Another factor the OPMD needs to take in consideration while planning and control a project is workforce behaviour. Parkinson's Law states "*work expands so as to fill the time available for its completion*" is been accurate especially with long period projects. The research here had considered the effect of schedule adherence on project completion time and its throughput; however, stress and project pressure might cause different behaviour in relation with workforce. Therefore, consideration to such facts related to human behaviours can increase the accuracy of the model (Gutierrez and Kouvelis 1991).

References:

Abdel-Hamid, T., Sengupta, K., and Ronan, D. (1993) Software Project Control: An Experimental Investigation of Judgment with Fallible Information. *IEEE Transactions on Software Engineering*, Vol. 19, Issue 6, pp. 603-612.

Abdulmalek, F., and Rajgopal, J. (2007) Analysing the Benefits of Lean Manufacturing and Value Stream Mapping via Simulation: A Process Sector Case Study. *International Journal of Production Economics*, Vol. 107, Issue 1, pp. 223-236.

Achanga, P., Shehab, E., Roy, R. and Nelder, G. (2006) Critical success factors for lean implementation within SMEs. *Journal of Manufacturing Technology Management*, Vol. 17, no. 4, pp. 460-471.

Ahmed, F. and Capretz, L. (2007) Managing the Business of Software Product Line: An Empirical Investigation of Key Business Factors. *Information and Software Technology*, Vol. 49, Issue 2, pp. 194-208.

Ahuja, V. and Thiruvengadam, V. (2004) Project scheduling and monitoring: current research status. *Construction Innovation: Information, Process, Management*, Vol. 4, no. 1, pp. 19 – 31.

Akao, Y. and Mazur, G. (2003) The Leading Edge in QFD: Past, Present and Future. *International Journal of Quality & Reliability Management*, Vol. 20, no. 1, pp. 20-35.

Alexander, L. and Davis, A. (1991) Criteria for Selecting Software Process Models. *IEEE, Proceedings of the 15th Annual International Conference of Computer Software and Application*, 11th-13th, Sep, 1991, Tokyo, Japan, pp.521- 528.

Alkaabi, M., Khalil, R. and Stockton, D. (2009) Implementing Lean in Software Development Operations. *Proceedings of the International Conference on Product Lifecycle Management-PLM-SP5*, 6th-8th July, 2009, Bath University, Bath, UK, pp. 690-698.

Alkaabi, M., Khalil, R. and Stockton, D. (2010) Improving Operations Management Planning and Control of a Service Project with Lean Principles. *Proceedings of the Junior Scientist Conference*, 4th-7th April, 2010, Vienna, Austria, pp. 61-62.

Allway, M. and Corbett, S. (2002) Shifting to Lean Service: Stealing a Page from Manufacturers' Playbooks. *Journal of Organisational Excellence*, Vol. 21, no. 2, pp. 45-54.

Amescua, A., Garcia, J., Velasco, M., Martinez, P., Ruiz, B., Llorens, J., Garcia, L., Calvo-Manzano, A. and Feliu, T. (2004) A Software Project Management Framework. *Journal of Information Systems Management*, Vol. 21, Issue 2, pp. 78-85.

Andersen, E., Grude, K. and Haug, T. (1995) The Global Directed Project Management. Kogan Page, 2nd edition, London.

Anderson, D. and Merna, T. (2003) Project Management Strategy: Project Management Represented as a Process Based Set of Management Domains and the Consequences for Project Management Strategy. *International Journal of Project Management*, Vol. 21, Issue 6, August 2003, pp. 387-393.

Andersson, C. and Runeson, P. (2007) A Spiral Process Model for Case Studies on Software Quality Monitoring- Method and Metrics. *Software Process Improvement and Practice*, Vol. 12, Issue 2, March/April 2007, pp. 125-140.

Anton, A. (2003) Successful Software Projects Need Requirement Planning. *Software, IEEE*, Vol. 20, Issue 3, pp. 45-46, ISSN: 0740-7459.

Antonioni, G., Spadoni, G. and Cozzani, V. (2007) A methodology for the quantitative risk assessment of major accidents triggered by seismic events. *Journal of Hazardous Materials*, Vol. 147, Issue 1-2, 17 August 2007, pp. 48–59.

Antony, J., Warwood, S., Fernandes, K. and Rowlands, H. (2001) Process Optimisation Using Taguchi Methods of Experimental Design. *MCB University Press*, Work Study, Vol. 50, no. 2, pp. 51-57, ISSN 0043-8022.

Aoyama, M. (1996) Beyond Software Factories: Concurrent-Development Process and an Evolution of Software Process Technology in Japan. *Information and Software Technology*, Vol. 38, Issue 3, pp. 133-143.

Apaiah, R., Hendrix, E., Meerdink, G. and Linnemann, A. (2005) Qualitative methodology for Efficient Food Chain Design. *Trends in Food Science & Technology*, Vol. 16, Issue 5, pp. 204–214.

Arbos, L. (2002) Design of a Rapid Response and High Efficiency Service by Lean Production Principles: Methodology and Evaluation of Variability of Performance. *International Journal of Production Economics*, Vol. 80, Issue 2, pp. 169-183.

Archibald, R. (1975) Managing High-Technology Programs and Projects. New York, Wiley.

Atkinson, R. (1999) Project Management: Cost, Time, and Quality, two best guesses and a Phenomenon, it's Time to Accept Other Success Criteria. *International Journal of Project Management*, Vol. 17, no. 6, pp. 337-342.

Aurich, J., Mannweiler, C. and Schweitzer, E. (2010) How to Design and Offer Services Successfully. *CIRP Journal of Manufacturing Science and Technology*, Vol. 2, Issue 3, pp. 136-143.

Baik, J., Boehm, B. and Steece, B. (2002) Disaggregating and Calibrating the CASE Tool Variables in COCOMO II. *IEEE Transactions on Software Engineering*, Vol. 28, Issue 11, pp. 1009-1022.

Ballard, G. (1999) Improving Work Flow Reliability. *Proceedings IGLC-7,* 26th-29th July, 1999, University of California, Berkeley, CA, USA, pp. 275-286.

Ballard, G. and Howell, G. (2003) Lean Project Management. *Building Research & Information*, Vol. 31, no. 2, pp. 119-133.

Balsamo, S., Marco, A., Inverardi, P. and Simeoni, M. (2004) Model-Based Performance Prediction in Software Development: A Survey. *IEEE Transactions on Software Engineering*, Vol. 30, Issue 5, pp. 295-310.

Bannerman, P. (2008) Risk and Risk Management in Software Projects: A Reassessment. *Journal of Systems and Software*, Vol. 81, Issue 12, pp. 2118-2133.

Barclay, C. and Osei-Bryson, K. (2010) Project Performance Development Framework: an Approach for Developing Performance Criteria & Measures for Information Systems (IS) Projects. *International Journal of Production Economics*, Vol. 124, Issue 1, pp. 272-292.

Barki, H., Rivard, S. and Talbot, J. (1993) Toward an Assessment of Software Development Risk, *Journal of Management Information Systems*, Vol. 10, no. 2, pp. 203-225.

Barnett, W. and Raja, M. (1995) Application of QFD to the Software Development Process. *International Journal of Quality & Reliability Management*, Vol. 12, no. 6, pp. 24-42.

Barney, S., Aurum, A. and Wohlin, C. (2008) A Product Management Challenge: Creating Software Product Value through Requirements Selection. *Journal of Systems Architecture*, Vol. 54, Issue 6, pp. 576-593.

Barros, M., Werner, C. and Travassor, G. (2004) Supporting Risks in Software Project Management. *The Journal of Systems and Software*, Vol. 70, Issue 1-2, pp. 21-35.

Ben-Menachem, M. (2008) Towards Management of Software as Assets: A Literature Review with Additional Sources. *Information and Software Technology*, Vol. 50, Issue 4, pp. 241-258.

Bennett, J., Bohoris, G., Aspinwall, E. and Hall, R. (1996) Risk Analysis Techniques and Their Application to Software Development. *European Journal of Operational Research*, Vol. 95, Issue 3, pp. 467-475.

Berander, P. and Wohlin, C. (2003) Identification of Key Factors in Software Process Management. *Proceedings of the 2003 International Symposium on Empirical Software Engineering (ISESE), IEEE*, 30th Sep-1st Oct, 2003, Bleking Institute of Technology, Ronneby, Sweden, pp. 316-325.

Berander, P., Khan, K. and Lehtola, L. (2006) Towards a Research Framework on Requirements Prioritization. *Proceedings the 6th Conference on Software Engineering Research and Practice*, 18th-19th Oct, 2006, Umea University, Umea, Sweden, pp. 39-48, ISSN 0348-0542.

Berkley, B. (1996) Analysing Service Blueprints Using Phase Distributions. *European Journal of Operational Research*, Vol. 88, Issue 1, pp. 152-164.

Bernstein, D., (1994) A Practitioner's Approach to Software Reliability. *Proceedings Annual Reliability and Maintainability Symposium*, 24th-27th Jan, 1994, Fairfax, VA, USA, pp. 349-358.

Bertrand, J. and Fransoo, J. (2002) Modelling and Simulation: Operations Management Research Methodologies using Quantitative Modelling. *International Journal of Operations & Production Management*, Vol. 22, no. 2, pp. 241-264.

Bhasin, S. (2008) Lean and Performance Measurement. *Journal of Manufacturing Technology Management*, Vol. 19, no. 5, pp. 670-684.

Bicheno, J. (2000) The Lean Toolbox, Buckingham, Picsie Books.

Bicheno, J. (2008) The Lean Toolbox for Service System. PICSE Books, Buckingham, UK.

Birk, A., (2002) Identifying and Managing Information Flows in Software Projects. *Proceedings of CONQUEST 2002 Conference on Quality Engineering in Software Technology*, VA, USA, pp. 1-12.

Blackstone, J. and Jonah, J. (2008), APICS Dictionary: Operations Management Body of Knowledge Framework, 12th Edition.

Boehm, B. (1981) Software Engineering Economics. Prentice-Hall Inc.

Boehm, B. (1984) Verifying and Validating Software Requirements and Design Specification. *Software IEEE*, Vol. 1, Issue 1, pp. 75-88.

Boehm, B. (1988) A Spiral Model of Software Development and Enhancement. *IEEE Computer*, Vol. 21, Issue 5, pp. 61-72.

Boehm, B. (1991) Software Risk Management: Principles and Practices. *IEEE Software*, Vol. 8, Issue 1, pp. 32-41.

Boehm, B. (2000) Safe and Simple Software Cost Analysis. *IEEE Transactions on Software*, Vol. 17, Issue 5, pp. 14-17.

Boehm, B. (2005) Value-Based SE: Seven Key Elements and Ethical Considerations. *Springer*, Berlin, Heidelberg, New York.

Boehm, B. and In, H. (1996) Identifying Quality-Requirement Conflicts. *IEEE Software*, Vol. 13, Issue 2, pp. 25-35.

Boehm, B. and Ross, R. (1988) Theory-W Software Project Management: a Case Study. *Proceedings the 10th International Conference on Software Engineering, IEEE*, 11th-15th April, 1988, California University, LA, CA, USA, pp. 30-40.

Boehm, B. and Ross, R. (1989) Theory-W Software Project Management: Principles and Examples. *IEEE Transactions on Software Engineering*, Vol. 15, no. 7, pp. 902-916.

Boehm, B., Abts, C. and Chulani, S. (2000b) Software Development Cost Estimation Approaches- A survey. *Annals of Software Engineering*, Vol. 10, no. 1-4, pp. 177-205.

Boggs, R. (2004) The SDLC and Six Sigma: an Essay Which is Which and Why?. *Issues on Information Systems*, Vol. V, no.1, pp.36-42.

Bolton, P. and Dewatripont, M. (1995) Information, Communication and Organisations: The Time and Budget Constraints of the Firm. *European Economic Review*, Vol. 39, pp. 691-699.

Bonner, J. (2005) The Influence of Formal Controls on Customer Interactivity in New Product Development. *Industrial Marketing Management*, Vol. 34, Issue 1, pp. 63-69.

Boot, E., Merrienboer, J. and Theunissen, N. (2008) Improving the Development of Instructional Software: Three Building-Block Solutions to Interrelate Design and Production. *Computers in Human Behaviour*, Vol. 24, Issue 3, pp. 1275-1292.

Bouchereau, V., Rowlands, H., (2000) Methods and Techniques to Help Quality Function Deployment (QFD). *Benchmarking: An International Journal*, Vol. 7, no. 1, pp. 8-19.

Bowen, D. and Youngdahl, W. (1998) Lean Service: in Defence of a Production-Line Approach. *International Journal of Service Industry Management*, Vol. 9, no. 3, pp. 207-255.

Brentani, U. (1991) Success Factors in Developing New Business Services. *European Journal of Marketing*, Vol. 25, no. 2, pp. 22-59.

Browning, T. (2000) Value-Based Product Development: Refocusing Lean. *Proceedings of the 2000 IEEE Engineering Management Society (EMS)*, 13th-15th August, 2000, Fort Worth, TX, USA, pp. 168-172.

Burgess, C., Dattani, I., Huges, **G., May, J. and Rees, K.,** (2001) Using Influence Diagrams to Aid the Management of Software Change. *Requirements Engineering Journal*, Vol. 6, no. 3, pp. 173-182.

Buyukozkan, G. and Feyzioglu, O. (2005) Group Decision Making to Better Respond Customer Needs in Software Development. *Computers & Industrial Engineering*, Vol. 48, Issue 2, pp. 427–441.

Caldwell, F. (2008) Organisational Engineering Management- Risk Intelligence: Applying KM to Information Risk Management. *VINE: the Journal of Information and Knowledge Management Systems*, Vol. 38, no. 2, pp. 163-166.

Camargo, J., Canzian, E., Almeida, J., Paz, S. and Basseto, B., (2001) Quantitative Analysis Methodology in Safety-Critical Microprocessor Applications. *Reliability Engineering and System Safety*, Vol. 74, Issue 1, pp. 53-62.

Canonico, P. and Soderlund, J. (2010) Getting Control of Multi-Project Organisation: Combining Contingent Control Mechanisms. *International Journal of Project Management*, Vol. 28, Issue 8, pp. 796-806.

Carbone, T. and Tippett, D., (2004) Project Risk Management Using the Project Risk FMEA. *Engineering Management Journal*, Vol. 16, no. 4, pp. 28-35.

Carino, C. (2006) Structural Layout Assessment by Orthogonal Array Based Simulation. *Mechanics Research Communications*, Vol. 33, Issue 3, pp. 292-301.

Carnevalli, J., Miguel, P. and Calarge, F. (2010) Axiomatic design application for minimising the difficulties of QFD usage. *Int. J. Production Economics*, Vol. 125, Issue 1, pp. 1–12.

Cervone, H. (2006) Managing Digital Libraries: the View from 30,000 Feet- Project Risk Management. *OCLC Systems & Services: International Digital Library Perspectives*, Vol. 22, no. 4, pp. 256-262.

Chan, L. and Wu, M. (2002) Quality Function Deployment: A Literature Review. *European Journal of Operational Research,* Vol. 143, Issue 3, pp. 463-497.

Charnes, A. and Cooper, W. (1961) Management models and industrial applications of linear programming. New York, Wiley.

Chatzigeorgiou, A. and Antoniadis, G. (2003) Efficient Management of Inspections in Software Development Projects. *Information and Software Technology*, Vol. 45, Issue 10, pp. 671-680.

Chen, M. and Wang, S. (2009) The critical factors of success for information service industry in developing international market: Using analytic hierarchy process (AHP) approach. *Expert Systems with Applications*, Vol. 37, Issue 1, pp. 694-704.

Cho, S. (2006) An Exploratory Project System for Eliciting Correlation Coefficient and Sequential Updating of Duration Estimation. *Expert Systems with Applications*, Vol. 30, Issue 4, pp. 553-560.

Chow, T. and Cao, D. (2008) A Survey Study of Critical Success Factors in Agile Software Projects. *The Journal of Systems and Software*, Vol. 81, Issue 6, pp. 961-971.

Chwif, L., Paul, R. and Barretoo, M. (2006) Discrete event simulation model reduction: A causal approach. *Simulation Modelling Practice and Theory*, Vol. 14, Issue 7, October 2006, Pages 930-944.

Cohen, M., and Palmer, G. (2004) Project Risk Identification and Management. *AACE International Transactions,* Vol. 1, no.2, pp. 1-5.

Collyer, S., and Warren, C. (2009) Project Management Approaches for Dynamic Environments. *International Journal of Project Management*, Vol. 27, Issue 4, pp. 355-364.

Costa, H., Barros, M. and Travassos, G. (2007) Evaluating Software Project Portfolio Risks. *The Journal of Systems and Software*, Vol. 80, Issue 1, pp.16-31.

Creswell, J. (2003) Research Design: Qualitative, Quantitative, and Mixed Methods Approaches. Sage Publications Inc., 2nd Edition.

Deephouse, C., Mukhopadhyay, T., Goldenson, R. and Kellner, M. (1995) Software Processes and Project Performance. *Journal of Management Information Systems*, Vol. 12, no. 3, pp. 187-205.

Dekleva, S. (1992) The Influence of the Information Systems Development Approach on Maintenance. *MIS Quarterly*, Vol. 16, no. 3, pp. 335-372.

Dey, P., Kinch, J. and Ogunlana, S. (2007) Managing Risk in Software Development Projects: A Case Study. *Industrial Management & Data Systems*, Vol. 107, no. 2, pp. 284-303.

Dey, P., Tabucanon, M. and Ogunlana, S. (1994) Planning for Project Control through Risk Analysis: a Case of Petroleum Pipeline Laying Project. *International Journal of Project Management*, Vol. 12, no. 1, pp. 23-33.

Dodin, B. and Elimam, A. (2008) Production, Manufacturing and Logistics - Integration of Equipment Planning and Project Scheduling. *European Journal of Operational Research*, Vol. 184, Issue 3, pp. 962-980.

Donaldson, S. and Siegel, S. (2007) Enriching Your Project Planning: Tying Risk Assessment to Resource Estimation. *IT Professional, IEEE*, Vol. 9, Issue 5, pp. 20-27.

Dorfman, M. (1997) Requirements Engineering. *IEEE Computer Society Press, Software Engineering*, Vol. 2, Issue 3/99, pp.7-22.

Duellmann, K., Kull, J. and Kunisch, M. (2010) Estimating asset correlations from stock prices or default rates—Which method is superior?. *Journal of Economic Dynamics & Control*, Vol. 34, Issue 11, pp. 2341–2357.

Dvir, D. and Lechler, T. (2004) Plans are Nothing, Changing Plans is everything: the Impact of Changes on Project Success. *Research Policy*, Vol. 33, Issue 1, pp. 1-15.

Dyba, T. (2003) Factors of Software Process Improvement Success in Small and Large Organisations: An Empirical Study in the Scandinavian Context. *ESEC/FSE'03, Proceedings of the 9th European Software Engineering Conference held jointly with ACM SIGSOFT International Symposium Foundations of Software Engineering*, 5th-7th Sep, 2003, New York, USA, pp. 148-157.

Ebert, C. (1999) Technical Controlling and Software Process Improvement. *The Journal of Systems and Software*, Vol. 46, Issue 1, pp. 25-39.

Ebert, C. (2007) The Impacts of Software Product Management. *The Journal of Systems and Software*, Vol. 80, Issue 6, pp. 850-861.

Ebert, C. and Man, J. (2008) Effectively Utilising Project, Product, and Process Knowledge. *Information and Software Technology*, Vol. 50, Issue 6, pp. 579-594.

Engel, A. and Last, M. (2007) Modelling Software Testing Costs and Risks Using Fuzzy Logic Paradigm. *The Journal of Systems and Software*, Vol. 80, Issue 6, pp. 817-835.

Enkel, E., Kausch, C. and Gassmann, O. (2005) Managing the Risk of Customer Integration. *European Management Journal*, Vol. 23, no. 2, pp. 203–213.

Erlang, A. (1909) The Theory of Probabilities and Telephone Conversations. *Nyt Tidsskrift for Matematik*, B 20.

Esaki, K., Yamada, S., Takahashi, M. and Hihara, K. (2002) A Quality Engineering Approach to Human Factors Affecting Software Reliability in Design Process. *Electronics and Communications in Japan*, Part 3, Vol. 85, no. 3, pp. 33-42.

Feather, M. (2004) Towards a Unified Approach to the Representation of, and Reasoning with, Probabilistic Risk Information about Software and its System Interface. *Proceeding of the 15th International Symposium on Software Reliability Engineering (ISSRE), IEEE*, 2nd-5th November, 2004, California Institute of Technology, Pasadena, CA, USA, pp. 391-402.

Fenton, N. and Neil, M. (1999) A Critique of Software Defect Prediction Models. *IEEE Transactions on Software Engineering*, Vol. 25, no. 5, pp. 675-689.

Ferreira, S., Collofello, J., Shunk, D. and Mackulak, G. (2009) Understanding the Effects of Requirements Volatility in Software Engineering by Using Analytical Modelling and Software Process Simulation. *The Journal of Systems and Software*, Vol. 82, Issue 1, pp. 1568–1577.

Fragniere, E., Gondzio, J. and Yang, X. (2010) Operations Risk Management by Optimally Planning the Qualified Workforce Capacity. *European Journal of Operational Research*, Vol. 202, Issue 2, pp. 518-527.

Fung, R., Popplewell, K. and Xie, J. (1998) An Intelligent Hybrid System for Customer Requirements Analysis and Product Attribute Targets Determination. *International Journal of Production Research*, Vol. 36, no. 1, pp. 13-348.

Furugaki, K., Takagi, T., Sakata, A., and Okayama, D. (2007) Innovation in Software Development Process by Introducing Toyota Production System. *Fujitsu Science Technology*, Vol. 43, Issue 1, pp.139-150.

Gantt, H. (1916) Work, Wages and Profits. 2nd Edition Engineering Magazine Co., New York.

Ghalayini, A. and Nobble, J. (1996) the Changing Basis of Performance Measurement. *International Journal of Operations &Production Management*, Vol. 16, no. 8, pp. 63-80.

Goldratt, E. and Cox, J. (1984) The Goal: A Process of On-going Improvement. North River Press, Corton-on-Hudon, New York.

Gong, B., Yen, D. and Chou, D. (1998) A Manager's Guide to Total Quality Software Design. *Industrial Management & Data Systems*, Vol. 93, no. 3, pp. 100-107.

Gonzalez, M., Quesada, G., Gourdin, K. and Hartley, M. (2008) Designing a Supply Chain Management Academic Curriculum Using QFD and Benchmarking. *Quality Assurance in Education*, Vol. 16, no. 1, pp. 36-60.

Gotterbarn, D. (2002) Reducing Software Failures: Addressing the Ethical Risks of the Software Development Lifecycle. *AJIS*, Vol. 9, no. 2, pp. 155-165.

Gould, L. (2006) PLM and Lean Product Development. *Automotive Design and Production*, Vol. 118, Issue 9, pp. 64-66.

Graves, T., Karr, A., Marron, J. and Siy, H. (2000) Predicting Fault Incidence using Software Change History. *IEEE Transactions on Software Engineering*, Vol. 26, no. 7, pp. 653-661.

Greasley, A. (2006) Operations Management. John Wiley & Sons Ltd, West Sussex.

Greer, D. and Ruhe, G. (2004) Software Release Planning: an Evolutionary and Iterative Approach. *Information and Software Technology*, Vol. 46, Issue 4, pp. 243-253.

Gunasekaran, A. and Yusuf, Y. (2002) Agile Manufacturing: a Taxonomy of Strategic and Technological Imperatives. *International Journal of Production Research*, Vol. 40, no. 6, pp. 1357-1385.

Guthrie, V. and Parikh, P. (2004) Software Safety Analysis: Using the Entire Risk Analysis Toolkit. *Proceedings of the Reliability and Maintainability (RAMS) Annual Symposium, IEEE*, 26th-29th Jan, 2004, Knoxville, TN, USA, pp. 272-279.

Gutierrez, G. and Kouvelis, P. (1991) Parkinson's Law and its Implications for Project Management. *Management Science*, Vol. 37, no. 8, pp. 990-1001.

Haag, S., Raja, M., and Schkade, L. (1996) Function Deployment: Usage in Software Development. *COMMUNICATIONS OF THE ACM*, Vol. 39, no. 1, pp. 41-49.

Han, W. and Huang, S. (2007) An Empirical Analysis of Risk Components and Performance on Software Projects. *The Journal of Systems and Software*, Vol. 80, Issue 1, pp. 42-50.

Hanna, A. and Gunduz, M. (2005) Early Warning Signs for Distressed Projects. Canadian Journal of Civil Engineering, Vol. 32, no. 5, *NRC*, Canada, pp.796-802.

Hans, E., Herroelen, W., Leus, R. and Wullink, G. (2007) A Hierarchical Approach to Multi-project Planning under Uncertainty. *Omega*, Vol. 35, Issue 5, pp. 563-577.

Hari, A., Kasser, J. and Weiss, M. (2007) How Lessons Learned from Using QFD Led to the Evolution of a Process for Creating Quality Requirements for Complex Systems. *Systems Engineering*, Vol. 10, no. 1, pp. 45-63.

Hartman, F. and Ashrafi, R. (2004) Development of the SMART Project Planning Framework. *International Journal of Project Management*, Vol. 22, Issue 6, pp. 499-510.

Hazzan, O. and Hadar, I. (2008) Why and how can human-related measures support software development processes?. *The Journal of Systems and Software*, Vol. 81, Issue 7, pp. 1248–1252.

He, Y., Chan, L. and Wu, M. (2007) Balancing Productivity and Consumer Satisfaction for Profitability: Statistical and Fuzzy Regression Analysis. *European Journal of Operational Research*, Vol. 176, Issue 1, pp. 252-263.

Head, R. (1984) Planning Techniques for Systems Management. QED Information Sciences, Inc., Wellesley Hills, Massachusetts.

Heiat, A. (2002) Comparison of Artificial Neural Network and Regression Models for Estimating Software Development Effort. *Information and Software Technology*, Vol. 44, Issue 15, pp. 911-922.

Herzwurm, G. and Schockert, S. (2003) The Leading Edge in QFD for Software and Electronic Business. *International Journal of Quality & Reliability Management*, Vol. 20, no. 1, pp. 36-55.

Herzwurm, G., Schockert, S. and Pietsch, W. (2003) QFD for Customer-Focused Requirements Engineering. *Proceedings of the 11th IEEE International Requirements Engineering Conference, IEEE*, 8th-12th September, 2003, Stuttgart University, Germany, pp. 330-338.

Hicks, B. (2007) Lean Information Management: Understanding and Eliminating Waste. *International Journal of Information Management*, Vol. 27, Issue 4, pp. 233-249.

Highsmith, J. and Cockburn, A. (2001) Agile Software Development: The Business of Innovation. *IEEE Computer*, Vol. 34, Issue 9, pp. 120-127.

Hines, P. and Rich, N. (1997) The seven value stream mapping tools. *International Journal of Operations and Production Management*, Vol. 17, no. 1, pp. 46-64.

Hines, P., Francis, M. and Found, P. (2006) Towards lean product lifecycle management A framework for new product development. *Journal of Manufacturing Technology Management*, Vol. 17, no. 7, pp. 866-887.

Hoch, D., Roeding, C., Purkert, G., Lindner, S. and Muller, R. (2000) Secrets of Software Success: Management Insights from 100 Software Firms around the World. *Harvard Business School Press*, Boston, MA.

Hollenbach, C., Young, R., Pflugrad, A. and Smith, D. (1997) Combining Quality and Software Improvement. *Communications of the ACM*, Vol. 40, no. 6, pp. 41-45.

Holmberg, L., Nilsson, A., Olsson, H. and Sandberg, A. (2008) Appreciative Inquiry in Software Process Improvement. *Software Process Improvement and Practice*, Vol. 14, Issue 2, pp. 107-125.

Hopp, W. and Spearman, M. (2000) Factory Physics: Foundation of Manufacturing Management., Irwin McGraw-Hill, 2nd Edition.

Horman, M. and Kenley, R. (1996) The Application of Lean Production to Project Management. *Proceedings of the 4th Annual Conference of the International Group for Lean Construction*, 6th-8th August, 1996, Birmingham, UK, pp. 1-8.

Huang, C. and Lo, J. (2006) Optimal Resource Allocation for Cost and Reliability of Modular Software Systems in the Testing Phase. *The Journal of Systems and Software*, Vol. 79, Issue 5, pp. 653-664.

Huang, S. and Han, W. (2008) Exploring the Relationship between Software Project Duration and Risk Exposure: a Cluster Analysis. *Information & Management*, Vol. 45, Issue 3, pp. 175-182.

Hughes, B. and Cotterell, M. (2006) Software Project Management. Berkshire, McGraw-Hill Education.

Ireland, L. (1997) Managing Multiple Project in twenty-first Century. *Managing Multiple Projects*, Marcel Dekker Inc., New York, pp. 21-34.

Jahangirian, M., Eldabi, T., Naseer, A., Stergioulas, L. and Young, T. (2010) Simulation in Manufacturing and Business: A Review. *European Journal of Operational Research*, Vol. 203, Issue 1, pp. 1-13.

Johnston, R. and Brennan, M. (1996) Planning or Organizing: the Implications of Theories of Activity for Management of Operations. *Omega, International Journal of Management*, Vol. 24, no. 4, pp. 367-384.

Johnston, R. and Clark, G. (2008) Service Operations Management: Improving Service Delivery. Prentice Hall, 3rd Edition.

Jones, C. (1996) Strategies for Managing Requirements Creep. *Computers IEEE*, Vol. 29, Issue 6, pp. 92-94.

Jonsson, P. and Wohlin, C. (2006) Using Checklists to Support the Change Control Process- a Case Study. *Proceedings the 6th Conference on Software Engineering Research and Practice*, 17th-18th October, 2006, Umea University, Umea, Sweden, pp. 71-80.

Kalpakjian, S. (1995) Manufacturing Engineering and Technology, Addison-Wesley Publishing Company, 3rd Edition.

Karlsson, J. (1997) Managing Software Requirements Using Quality Function Deployment. *Software Quality Journal*, Vol. 6, no.4, pp. 311-325.

Karlsson, J., Wohlin, C. and Regnell, B. (1998) An Evaluation of Methods for Prioritising Software Requirements. *Information and Software Technology*, Vol. 39, Issues 14-15, pp. 939-947.

Karmarkar, U. and Pitbladdo, R. (1995) Service Markets and Competition. *Journal of Operations Management*, Vol. 12, Issues 3-4, pp. 397-411.

Kastor, A. and Sirakoulis, K. (2009) The Effectiveness of Resource Levelling Tools for Resource Constraint Project Scheduling Problem. *International Journal of Project Management*, Vol. 27, Issue 5, pp. 493-500.

Kasunic, M. (2008) A Data Specification for Software Project Performance Measures: Results of a Collaboration on Performance Measurement. *Software Engineering Institute, Software Engineering Process Management*, July 2008, pp. 22-56.

Kececi, N. and Modarres, M., (1998) Software Development Life Cycle Model to Ensure Software Quality. *Proceedings of the 4th International Conference on PSAM*, 21st-23rd Oct, 1998, New York, USA, pp. 1-6.

Keil, M., Cule, P., Lyytinen, K. and Schmidt, R. (1998) A Framework for Identifying Software Project Risks. *Communications of the ACM*, Vol. 41, no. 11, pp. 76-83.

Kerzner, H. (2006) Project Management: A Systems Approach to Planning, Scheduling and Controlling. New York: Wiley, 9th Edition.

Khalil, R. (2005) Predicting the Effect of Variability on the Efficiency of Flow Processing Systems. PhD Thesis.

Khalil, R., Stockton, D. and Steeple, D. (2010) Predicting the Effects of Cycle Time Variability on the Efficiency of Electronics Assembly Mixed-model, Zero-buffer Flow Processing Lines. *International Journal of Computer Integrated Manufacturing*, accepted for publication.

Khalil, R., Stockton, D., Newman, S. and Ardon-Finch, J. (2006) Designing Multi-Component Flexible Manpower Lines. *Transactions of society of automotive engineers of Japan*, Vol. 37, no. 1, pp. 141-146.

Kilpatrick, J. (2003) Lean Principles. *Utah Manufacturing Extension Partnership Organisation*, Vol. 1, no. 2, pp. 1-5.

Kim, Y., Park, K., Ko, J. (2003) A Symbiotic Evolutionary Algorithm for the Integration of Process Planning and Job Shop Scheduling. *Computers & Operations Research*, Vol. 30, Issue 8, pp. 1151-1171.

Kindler, N., Krishnakanthan, K. and Tinaikar, R. (2007) Applying Lean to Application Development and Maintenance. *McKinsey & Company*, Vol. 2, no. 1, pp. 1-11.

Knod, E. and Schonberger, R. (2001) Operations Management: Meeting Customers' Demands. McGraw-Hill Higher Education, Division of the McGraw-Hill Companies, 7th Edition.

Kosuge, R., Holm, M., Modig, N. and Ahlstrom, P. (2009) Adoption of the Lean Concept at a Toyota Car Dealer: Identifying the Key Factors. *Euroma*, Vol. 24, no. 3, pp. 1-10.

Kouskouras, K. and Georgiou, A. (2007) A Discrete Event Simulation Model in the Case of Managing a Software Project. *European Journal of Operational Research*, Vol. 181, Issue 1, pp. 374-389.

Krasna, M. and Rozman, I. (1996) Management of Software Development Processes. *Proceedings of the International Conference on Engineering and Technology Management, IEEE*, 18th-20th August, 1996, Vancouver, BA, Canada, pp. 253-257.

Kulk, G. and Verhoef, C. (2008) Quantifying Requirements Volatility Effects. *Science of Computer Programming*, Vol. 72, Issue 1, pp. 136-175.

Kumar, D. (2005) Lean Software Development. *The Project Perfect White Paper Collection*, Vol. 2, no. 1, pp. 1-8.

Lai, X., Xie, M., Tan, K. and Yang, B., (2008) Ranking of customer requirements in a competitive environment. *Computers & Industrial Engineering*, Vol. 54, Issue 2, pp. 202-214.

Lancaster, J. and Cheng, K. (2008) Project Schedule Optimisation Using a Genetic Algorithm Approach. *AACE International Transactions*, PS.S04, pp.4.1-4.10.

Laslo, Z. (2010) Project Portfolio Management: an Integrated Method for Resource Planning and Scheduling to Minimise Planning/Scheduling-Dependent Expenses. *International Journal of Project Management*, Vol. 28, Issue 6, pp. 609-618.

Laszlo, G. (1999) Project Management: A Quality Management Approach. *The TQM Magazine*, Vol. 11, no. 3, pp. 157-160.

Lauesen, S. (2003) Task Descriptions as Functional Requirements. *IEEE Software*, Vol. 20, Issue 2, pp.58-65.

Lee, H. and Tang, C. (1997) Modelling the Cost and Benefits of Delayed Product Differentiation. *Management Science*, Vol. 43, no. 1, pp. 40-53.

Lee, H. and Tang, C. (1998) Variability Reduction through Operations Reversal. *Management Science*, Vol. 44, no. 2, pp. 162-172.

Lee, J. and Lee, N. (2006) Least Modification Principle for Case-Based Reasoning: a Software Project Planning Experience. *Expert Systems with Applications*, Vol. 30, Issue 2, pp. 190-202.

Lee, S. and Schniederjans, M. (1994) Operations Management. Boston Toronto, Houghton Mifflin Company.

Lee-Mortimer, A. (2006) A Lean Route to Manufacturing Survival, *Assembly Automation*, Vol. 26, no. 4, pp. 265-272.

Lehtonen, J., Holmstrom, J. and Slotte, J. (1999) Constraints to Quick Response Systems in the Implosive Industries. *Supply Chain Management*, Vol. 4, no. 1, pp. 51-57.

Lewis, J. (2001) Project Planning Scheduling and Control. McGraw-Hill, 3rd Edition.

Liker, J. (2004) The Toyota Way: 14 Management Principles from the World's Greatest Manufacturer. New York, McGraw-Hill.

Lima, E., Costa, S. and Faria, A. (2009) Taking Operations Strategy into Practice: Developing a Process for Defining Priorities and Performance Measures. *International Journal of Production Economics*, Vol. 122, Issue 1, pp. 403-418.

Lin, C., Abdel-Hamid, T. and Sherif, J. (1997) Software Engineering Process Simulation Model. *Journal of Systems Software*, Vol. 38, Issue 3, pp. 263-277.

Liu, F., Noguchi, K., Dhungana, A., Srirangam, V. and Inuganti, P. (2006) A Quantitative Approach for Setting Technical Targets Based on Impact Analysis in Software Quality Function Deployment (SQFD). *Software Quality Journal*, Vol. 14, no. 2, pp. 113-134.

Liu, J., Chen, V., Chan, C. and Lie, T. (2008) The Impact of Software Process Standardization on Software Flexibility and Project Management Performance: Control Theory Perspective. *Information and Software Technology*, Vol. 50, Issues 9-10, pp. 889-896.

Ljungberg, A. (2002) Process Measurement. *International Journal of Physical Distribution & Logistics Management*, Vol. 32, no. 4, pp. 254-287.

Looy, B., Gemmel, P. and Dierdonck, R. (2003) Services Management: an Integrated Approach. Prentice Hall, 2nd Edition.

Luo, X., Kwong, C. and Tang, J. (2010) Determining Optimal Levels of Engineering Characteristics in Quality Function Deployment under Multi-Segment Market. *Computers & Industrial Engineering*, Vol. 59, Issue 1, pp. 126-135.

Lyytinen, K., Mathiassen, L. and Ropponen, J. (1998) Attention Shaping and Software Risk- A Categorical Analysis of Four Classical Risk Management Approaches. *Information Systems Research*, Vol. 9, no. 3, pp. 233-255.

Machuca, J., Gonzalez-Zamora, M. and Aguilar-Escobar, V. (2007) Service Operations Management Research. *Journal of Operations Management*, Vol. 25, Issue 3, pp. 585-603.

Maqsood, M. and Javed, T. (2007) Practicum in Software Project Management: an Endeavour to Effective and Pragmatic Software Project Management Education. *Proceeding ESEC-FSE Companion '07 and the 6th Joint Meeting on European Software Engineering Conference and the ACM SIGSOFT Symposium on the foundations of Software Engineering. ESEC/FSE*, 3rd-7th Sep, 2007, Cavat near Dubrovnik, Croatia, pp. 471-479, ISBN: 978-1-59593-812-1.

Martine, C. (2001) Teaming the Service Sector. *Team Performance Management: An International Journal*, Vol. 7, Issue ½, ISSN 1352-7592.

Mateosian, R. (2003) Managing Software Projects. *IEEE Micro*, Vol. 23, Issue 4, pp. 11-13.

Maxwell, K. and Forselius, P. (2000) Benchmarking Software Development Productivity. *IEEE Software*, Vol. 17, Issue 1, pp. 80-88.

McBride, T. (2008) The Mechanisms of Project Management of Software Development. *The Journal of Systems and Software*, Vol. 81, Issue 12, pp. 2386-2395.

McConnell, S. (1996) Rapid Development: Taming Wild Software Schedules. and Microsoft Press, Redmond, Washington, ISBN 1556159005.

McConnell, S. (1997) Achieving Leaner Software. *IEEE Software*, Vol. 14, Issue 6, pp. 127-128.

McFarlan, F. (1981) Portfolio Approach to Information Systems. *Harvard Business Review*, Vol. 59, no. 5, pp. 142-150.

McManus, J. and Wood-Harper, T. (2007) Software Engineering: A Quality Management Perspective. *The TQM Magazine*, Vol.19, no.4, pp. 315-327.

Mehta, M., Anderson, D. and Raffo, D. (2008) Providing Value to Customers in Software Development through Lean Principles. *Software Process Improvement and Practice*, Vol. 13, Issue 1, pp. 101-109.

Meredith, J. (1992) The Management of Operations: A Conceptual Emphasis. John Wily & Sons 4th Edition.

Mohan, K. and Jain, R. (2008) Using Traceability to Mitigate Cognitive Biases in Software Development. *Communications of the ACM*, Vol. 51, no. 9, pp. 110-114.

Molokken-Ostvold, K. and Jorgensen, M. (2005) A Comparison of Software Project Overruns- Flexible versus Sequential Development Models. *IEEE Transactions on Software Engineering*, Vol. 31, no. 9, pp. 754-766.

Morgenshtern, O., Raz, T. and Dvir, D. (2007) Factors Affecting Duration and Effort Estimation Errors in Software Development Projects. *Information and Software Technology*, Vol. 49, Issue 8, pp. 827-837.

Morien, R. (2005) Agile Management and the Toyota Way for Software Project Management. *Proceedings the 3rd IEEE International Conference on Industrial Informatics*, 10th-12th August, 2005, Curtin University of Technology, Perth, WA, Australia, pp. 516-522.

Mota, C., Almeida, A. and Alencar, L. (2009) A Multiple Criteria Decision Model for Assigning Priorities to Activities in Project Management. *International Journal of Project Management*, Vol. 27, Issue 2, pp. 175-181.

Munson, J. (1996) Software Faults, Software Failures, and Software Reliability Modelling. *Information and Software Technology*, Vol. 38, Issue 11, pp. 687-699.

Murphy, A. and Ledwith, A. (2007) Project Management Tools and Techniques in High-Technology SMEs. *Management Research News*, Vol. 30, no. 2, pp. 153-166.

Narasimhan, R. and Jayaram, J. (1998) Reengineering Service Operations: a Longitudinal Case Study. *Journal of Operations Management*, Vol. 17, Issue 1, pp. 7-22.

Nguyen, T., Marmier, F. and Gourc, D., (2010) A Decision-making Tool to Maximise Chances of Meeting Project Commitments. *International Journal of Production Economics*, PII, pp. 1-11, doi:10.1016/j.ijpe.2010.11.023.

Nuseibeh, B., Easterbrook, S. and Russo, A. (2001) Making Inconsistency Respectable in Software Development. *The Journal of Systems and Software*, Vol. 58, Issue 2, pp. 171-180.

O'Regan, N. and Ghobadian, A. (2002) Formal Strategic Planning: the Key to Effective Business Process Management. *Business Process Management Journal*, Vol. 8, no. 5, pp. 416-429.

Ozdamar, L. and Alanya, E. (2001) Uncertainty Modelling in Software Development Projects. *Annals of Operations Research*, Vol. 102, no. 1-4, pp. 157-178.

Pai, W. (2002) A Quality-Enhancing Software Function Deployment Model. *Information Systems Management*, Vol. 9, no. 3, pp. 20-24.

Pang, A., Cropp, F. and Cameron, G. (2006) Corporate crisis planning: tensions, issues, and contradictions. *Journal of Communication Management*, Vol. 10, no. 4, pp. 371-389.

Parnas, D. and Clements, P. (1986) A Rational Design Process: How and Why to Fake It?. *IEEE Transactions on Software Engineering*, Vol. SE-12, no. 2, pp. 251-256.

Parnell-Klabo, E. (2006) Introducing Lean Principles with Agile Practices at a Fortune 500 Company. *Proceedings of Agile 2006 Conference*, 23rd-28th July, 2006, Minneapolis, MN, USA, pp. 242-249.

Parry, G., Mills, J. and Turner, C. (2010) Lean Competence: Integration of Theories in Operations Management Practice. *Supply Chain Management: An International Journal*, Vol. 15, no. 3, pp. 216-226.

Patanakul, P. and Milosevic, D. (2009) The Effectiveness in Managing a Group of Multiple Projects: Factors of Influence and Measurement Criteria. *International Journal of Project Management*, Vol. 27, Issue 3, pp. 216-233.

Pavur, R., Jayakumar, M. and Clayton, H. (1999) Software Testing Metrics: Do They Have Merit?. *Industrial Management & Data Systems*, Vol. 99, Issue 1, pp. 5-10.

Pedler, M. and Abbott, C. (2008) Lean and Learning: Action Learning for Service Improvement. *Leadership in Health Services*, Vol. 21, no. 2, pp. 87-98.

Perera, G. and Fernando, M. (2007) Enhanced Agile Software Development – Hybrid Paradigm with Lean Practice. *Proceedings 2nd International Conference on Industrial and Information Systems, IEEE*, 9th-11th August, 2007, Penadeniya, Sri Lanka, pp. 239-244.

Perminova, O., Gustafsson, M. and Wikstrom, K. (2008) Defining Uncertainty in Projects: a New Perspective. *International Journal of Project Management*, Vol. 26, Issue 1, pp. 73-79.

Petter, S. (2008) Managing User Expectations on Software Projects: Lessons from the Trenches. *International Journal of Project Management*, Vol. 26, Issue 7, pp. 700–712.

Pfahl, D. and Lebsanft, K., (2000) Using Simulation to Analyse the Impact of Software Requirement Volatility on Project Performance. *Information and Software Technology*, Vol. 42, Issue 14, pp. 1001-1008.

Philpott, E., Hamblin, D., Baines, T. and Kay, G. (2004) The Use of Models and Methods for Strategic Planning: Towards a Holistic View of Strategy. *International Transactions in Operational Research*, Vol. 11, Issue 2, pp. 203-216.

Piercy, N. and Rich, N. (2009) Lean Transformation in the Pure Service Environment: the Case of the Call Service Centre. *International Journal of Operations & Production Management*, Vol. 29, no. 1, pp. 54-76.

Plogret, K. (1996) The Tailoring Process in the German V-Model. *Journal of System Architecture*, Vol. 42, Issue 8, pp. 601-609.

Pollack, J. (2007) The Changing Paradigms of Project Management. *International Journal of Project Management*, Vol. 25, Issue 3, pp. 266-274.

Poppendieck, M. (2007) Lean Software Development. *Proceedings the 29th International Conference on Software Engineering*, Washington DC, USA, pp. 165-166, ISBN:0-7695-2892-9.

Poppendieck, M. and Poppendieck, T. (2007) Implementing Lean Software Development: From Concept to Cash. Addison-Wesley.

Prasad, B. (1998) Review of QFD and Related Deployment Techniques. *Journal of Manufacturing Systems*, Vol. 17, no. 3, pp. 221-234.

Pretschner, A., Lotzbeyer, H. and Philipps, J. (2003) Model Based Testing in Incremental System Development. *the Journal of Systems and Software*, Vol. 70, Issue 3, pp. 315-329.

Proctor, T. and Doukakis, I. (2003) Change Management: the Role of Internal Communications and Employee Development. *Corporate Communications: An Internal Journal*, Vol. 8, no. 4, pp. 268-277, ISSN 1356-3289.

Project Management Institute (2004) A Guide to the Project Management Body of Knowledge. Project Management Institute, ISBN: 1-930699-45-X, 3rd Edition.

Pyhajarvi, M., Rautiainen, K. and Ithonen, J. (2002) Increasing Understanding of the Modern Testing Perspective in Software Product Development Project. *Proceedings of the 36th Annual Hawaii International Conference on System Science*, *Big Island, Hawaii, 06th-09th January, 2002*, pp. 1-10.

Raman, S. (1998) Lean Software Development: Is It Feasible?. *Proceedings the 17th Digital Avionics Systems Conference, Bellevue*, 31st Oct- 7th Nov, 1998, WA, USA, Vol. 1, pp. C13/1- C13/18.

Ramli, A., Watada, J. and Pedrycz, W. (2011) Real-time fuzzy regression analysis: A convex hull approach. *European Journal of Operational Research*, Vol. 210, Issue 3, pp. 606–617.

Ranky, P. (2007) Eighten 'monozukuri-focused' Assembly Line Design and Visual Factory Management Principles with DENSO Industrial Examples. *Assembly Automation*, Vol. 27, no. 1, pp. 12-16.

Rautiainen, K., Vuornos, L. and Lassenius, C. (2003) An Experience in Combining Flexibility and Control in a Small Company's Software Product Development Process. *Proceedings of the 2003 International Symposium on Empirical Software Engineering (ISESE'03), IEEE*, Espoo, Finland, 30th Sept- 1st Oct, 2003, pp. 1-10.

Reel, J. (1999) Critical Success Factors in Software Projects. *IEEE Software*, Vol. 16, Issue 3, pp. 19-23, ISSN: 0740-7459.

Reinertsen, D. (2005) How Lean Product Development Sparked a Revolution. *Industrial Engineer*, Vol. 2, no. 1, pp. 40-45.

Reinertsen, D. (2006) Eliminating Development Queues. *Emerald Group Publishing Limited, Strategic Direction*, Vol. 22, no. 7, pp. 33-35.

Reinertsen, D. and Shaeffer, L. (2005) Making R&D Lean. *Research Technology Management*, Vol. 48, no. 4, pp. 51-57.

Rexfelt, O. and Rosenblad, E. (2006) The Progress of User Requirements Through a Software Development Project. *International Journal of Industrial Ergonomics*, Vol. 36 Issue 1, pp. 73-81.

Riezebos, J., Klingenberg, W. and Kicks, C. (2009) Lean Production and Information Technology: Connection or Contradiction?. *Computers in Industry*, Vol. 60, Issue 4, pp. 237-247.

Robinson, S. (2002) General Concepts of Quality for Discrete-event Simulation. *European Journal of Operational Research*, Vol. 138, Issue 1, pp. 103-117.

Ronconi, D. and Henriques, L. (2009) Some Heuristic Algorithms for Total Tardiness Minimization in a Flow shop with Blocking. *Omega*, Vol. 37, Issue 2, pp. 272-281.

Ropponen, J. and Lyytinen, K. (2000) Components of Software Development Risk: How to Address Them? A Project Manager Survey. *IEEE Software*, Vol. 26, no. 2, pp. 98-112.

Rose, J., Pedersen, K., Hosbond, J. and Kraemmergaard, P., (2007) Management Competences, not Tools and Techiques: a Grounded Examination of Software Project Management at WM-data. *Information and Software Technology*, Vol. 49, Issue 6, pp. 605-624.

Roy, P. (2007) Self-Management and the Future of Software Design. *Proceedings of the 3rd International Workshop on Formal Aspects of Component Software (FACS 2006), Electronics Notes in Theoretical Computer Science*, Louvain-la-Neuve, Belgium, Vol. 182, pp. 201-217.

Royce, W. (1970) Managing the Development of Large Software Systems. *Proceedings of IEEE WESCON, August 1970, The Institute of Electrical and Electronics Engineers*, pp. 329-338.

Rozum, J. (1994) Defining and Understanding Software Measurement Data. *Proceeding the 5th International Applications of Software Measurements Conference*, Camegie Mellon University, Pittsburgh, PA, USA, pp.1-11.

Ruch, W., Fearon, H. and Wieters, C. (1992) Fundamentals of Production/Operations Management. 5th Edition, West Publishing Company.

Ruiz-Gonzalez, F. and Canfora, G. (2004) Software Process: Characteristics, Technology and Environments. *UPGRADE*, Vol. V, no. 5, pp. 6-10.

Ruuska, I., Ahola, T., Artto, K., Locatelli, G. and Mancini, M. (2010) A new governance approach for multi-firm projects: Lessons from Olkiluoto 3 and Flamanville 3 nuclear power plant projects. *International Journal of Project Management*, Vol. 29, Issue 6, pp. 1-14.

Salem, A., Rekabb, K. and Whittaker, J. (2004) Prediction of Software Failures through Logistic Regression. *Information and Software Technology*, Vol. 46, Issue 12, pp. 781-789.

Sawyer, S. and Guinan, J. (1998) Software Development: Processes and Performance. *IBM Systems Journal*, Vol. 37, Issue 4, pp. 552-568.

Schonsleben, P. (2007) Techniques for Planning and Control Dependent on Different Types of Flexibility. *Annals of the CIRP*, Vol. 56, no. 1, pp. 451-454.

Setamanit, S., Wakeland, W. and Raffo, D. (2007) Improving Global Software Development Project Performance Using Simulation. *Proceedings PICMET 2007*, 5th-9th August, 2007, Portland, Oregon, USA, pp. 2459-2466.

Shinkle, G. (2005) In Search of Lean Management. *IEE Manufacturing Engineer*, Vol. 12, no. 1, pp. 44-47.

Silver, E. (2004) Process Management Instead of Operations Management. *Manufacturing & Service Operations Management*, Vol. 6, no. 4, pp. 273-279.

Silverman, D. (2005) Doing Qualitative Research: a Practical Handbook. Sage, London, 2nd Edition.

Skinner, W. (1986) the Productivity Paradox. *Harvard Business Review*, Vol. 64, July-August, pp. 55-9.

Smith, P. and Reinertsen, D. (1992) Shortening the Product Development Cycle. *Research-Technology Management*, Vol. 3, no. 1, pp. 44-49.

Sobek II, D., Ward, A. and Liker, J. (1999) Toyota's Principles of Set-Based Concurrent Engineering. *Salon Management Review*, Vol. 40, no. 2, pp. 67-83.

Solingen, R. (2004) Measuring the ROI of Software Process Improvement. *IEEE Software*, Vol. 21, Issue 3, pp. 32-38.

Spear, S. (2006) Learning to Lead at Toyota. *Harvard Business Review*, Product 2890, pp. 1-11.

Spohrer, J., Maglio, P., Bailey, J. and Gruhl, D., (2007) Steps toward a Science of Service Systems. *IEEE Computer, Digital Object Identifier*, Vol. 14, Issue 1, pp. 71-77.

Staats, B. and Upton, D. (2009) Lean Principles, Learning, and Software Production: Evidence from Indian Software Services. *Harvard Business School Technology & Operations Management*, Unit Working Paper, no. 08-001, pp. 1-32.

States of Texas, Department of Information Resources, (2008) Generic Software Project Risk Factors. *DIR*.

Stephen, M. and Bates, P. (1993) Controlling Prototyping and Evolutionary Development. *Proceedings 4th International Workshop on Rapid System Prototyping*, 28th-30th June, 1993, Research Triangle Park, NC, USA, pp. 164-185.

Stevenson, W. (2005) Operations Management. McGraw-Hill Irwin, 8th Edition.

Stuart, F. and Tax, S. (1996) Planning for Service Quality: an Integrative Approach. *International Journal of Service Industry Management*, Vol. 7, no. 4, pp. 58-77.

Suardi, L. (2004) How to Manage Your Software Product Life Cycle with MAUI. *Communications of the ACM*, Vol. 47, no. 3, pp. 89-94.

Sukthomya, W. and Tannock, J., (2005) Taguchi experimental design for manufacturing process optimisation using historical data and a neural network process model. *International Journal of Quality & Reliability Management*, Vol. 22, no. 5, pp. 485-502.

Sutherland, J., Jakobsen, C. and Johnson, K., (2008) Scrum and CMMI Level 5: The Magic Potion for Code Warriors. *Proceedings of the 41st Hawaii International Conference on System Sciences*, 7th-10th January, 2008, Waikoloa, HI, pp. 466-474.

Sutton, J. (1996) Lean Software for the Lean Aircraft. *IEEE 15th Digital Avionics Systems Conference*, 27th-31st Oct, 1996, Marietta, GA, pp. 49-54.

Sutton, J. (2008) Welcoming Software into the Industrial Fold. *Cross Talk, the Journal of Defence Software Engineering*, Vol. 3, no. 1, pp. 11-15.

Swank, C. (2003) The Lean Service Machine. *Harvard Business Review*, Vol. 81, Issue 10, pp. 123-129.

Taguchi, G. (1987), System of Experimental Design: Engineering Methods to optimize quality and minimize costs. *Unipub/Kraus International Publication 2V*, New York.

Tahir, A., Darton, R. (2010) The Process Analysis Method of Selecting Indicators to Quantify the Sustainability Performance of a Business Operation. *Journal of Cleaner Production*, Vol. 18, Issues 16-17, pp. 1598-1607.

Tausworthe, R. (1992) Information Models of Software Productivity: Limits on Productivity Growth. *Journal of System Software*, Vol. 19, no. 2, pp. 185-201.

Taylor, F. (1911) Principles of Scientific Management. Reprinted (1998) by Dover Publications, Mineola, NY.

Taylor, L. (1999) A simulation study of WIP inventory drive systems and their effect on financial measurements. *Journal of Integrated Manufacturing Systems*, Vol. 10, no. 5, pp. 306-315.

Tchankova, L. (2002) Risk Identification – Basic Stage in Risk Management. *Environmental Management and Health*, Vol. 13, no. 3, pp. 290-297.

Tenenberg, J. (2008) An Institutional Analysis of Software Teams. *International Journal of Human-Computer Studies*, Vol. 66, Issue 7, pp. 484-494.

Tetnowski, J. and Damico, J. (2001) A demonstration of the advantages of qualitative methodologies in stuttering research. *Journal of Fluency Disorders*, Vol. 26, Issue 1, pp. 17-42**.**

Thiagarajan, S., Rajendran C. (2005) Scheduling in Dynamic Assembly Job-Shops to Minimize the Sum of Weighted Earliness, Weighted Tardiness and Weighted Flow time of Jobs. *Computers & Industrial Engineering*, Vol. 49, Issue 4, pp. 463-503.

Tiwana, A. (2004) An Empirical Study of the Effect of Knowledge Integration on Software Development Performance. *Information and Software Technology*, Vol. 46, Issue 13, pp. 899-906.

Toni, A. and Tonchia, S. (1996) Lean organization, management by process and performance measurement. *International Journal of Operations & Production Management*, Vol. 16, no. 2, pp. 221-236.

Tonini, A., Laurindo, F. and Spinola, S. (2007) An Application of Six Sigma with Lean Production Practices for Identifying Common Causes of Software Process Variability. *Proceedings of the PICMET 2007 Conference*, 5th-9th August, 2007, Portland, OR, pp. 2482-2490.

Tsai, H., Moskowitz, H. and Lee, L. (2003) Human Resource Selection for Software Development Projects using Taguchi's Parameter Design. *European Journal of Operational Research*, Vol. 151, Issue 1, pp.167-180.

Tunali, S. and Batmaz, I. (2003) Stochastic and Statistics: a Meta modelling methodology involving both qualitative and quantitative input factors. *European Journal of Operational Research*, Vol. 150, no. 1, pp. 437–450.

Turner, J. (1993) The Handbook of Project-based Management. McGraw-Hill, London.

Tyagi, R. and Das, C. (1999) Grouping Customers for Better Allocation of Resources to Serve Correlated Demands. *Computers & Operations Research*, Vol. 26, Issues 10-11, pp. 1041-1058.

Verner, J. and Evanco, W. (2005) In-House Software Development: What Project Management Practices Lead to Success?. *IEEE Software*, Vol. 22, Issue 1, pp. 86-93.

Vidal, L. and Marle, F. (2008) Understanding Project Complexity: Implications on Project Management. *Kybernetes*, Vol. 37, no. 8, pp. 1094-1110.

Vliet, H. (1993) Software Engineering: Principles and Practice. John Wiley & Sons Ltd, Chichester.

Wagner, K. and Durr, W. (2006) A Five-Step for Value-Based Planning and Monitoring of Systems Engineering Projects. *Proceedings of the 32nd EUROMICRO Conference on Software Engineering and Advanced Applications IEEE Transaction*, 29th Aug-1st Sep, 2006, Vienna, Austria, , pp. 1-9.

Wagner, M., Bhadury, J. and Peng, S. (2009) Risk Management in Un-Capacitated Facility Location Models with Random Demands. *Computers & Operations Research*, Vol. 36, Issue 4, pp. 1002-1011.

Wallace, L. and Keil, M., (2004) Software Project Risks and Their Effect on Outcomes. *Communication of the ACM*, Vol. 47, no. 4, pp. 68-73.

Wallace, L., Keil, M. and Rai, A., (2004) How Software Project Affects Project Performance: An Investigation of the Dimensions of Risk and Exploratory Model. *Decision Science*, Vol. 35, no. 2, pp. 289-321.

Wang, J. (1999) Fuzzy Outranking Approach to Prioritise Design Requirements in Quality Function Deployment. *International Journal of Production Research,* Vol. 37, no. 4, pp. 899-916.

Wang, T. and Huang, C. (2008) Optimizing Back-Propagation Networks via a Calibrated Heuristic Algorithm with an Orthogonal Array. *Expert Systems with Applications,* Vol. 34, Issue 3, pp. 1630-1641.

Weglarz, J., Jozefowska, J., Mika, M. and Waligora, G. (2011) Project Scheduling with Finite or Infinite Number of Activity Processing Modes- A Survey. *European Journal of Operational Research,* Vol. 208, Issue 3, pp. 177-205.

White, D. and Fortune, J. (2002) Current Practice in Project Management: An Empirical Study. *International Journal of Project Management*, Vol. 20, Issue 1, pp. 1-11.

Whittaker, B. (1999) What Went Wrong? Unsuccessful Information Technology Projects. *Information Management & Computer Security*, Vol. 7, no. 1, pp. 23-29.

Wikstrom, K., Hellsotrom, M., Artto, K., Kujala, J., Jujala, and Jujala, S. (2009) Services in Project-based Firms – Four Types of Business Logic. *International Journal of Project Management*, Vol. 27, Issue 2, pp. 113-122.

Williams, H. (1985) Model building in mathematical programming. New York, Wiley.

Williams, L. and Cockburn, A. (2003) Agile Software Development: It's about Feedback and Change. *IEEE Computer Society*, Vol. 36, Issue 6, pp. 39-40.

Williams, R., Bertsch, B., Wiele, T., Iwaarden, J., Smith, M. and Visser, R. (2006) Quality and Risk Management: What are the Key Issues?. *The TQM Magazine*, Vol. 18, no. 1, pp. 67-86.

Wirth, N. (1995) A Plea for Lean Software. *IEEE Computer,* Vol. 28, Issue 2, pp. 64-88.

Wohlgemuth, V., Page, B. and Kreutzer, W. (2006) Combining discrete event simulation and material flow analysis in a component-based approach to industrial environmental protection. *Environmental Modelling & Software,* Vol. 21, Issue 11, pp. 1607-1617.

Womack, J. and Jones, D. (1996) From Lean Production to the Lean Enterprise. *IEEE Engineering Management Review*, Vol. 24, no. 4, pp. 38-46.

Womack, J. and Jones, D. (2003) Lean Thinking: Banish Waste and Create Wealth in Your Corporation. Simon & Schuster.

Womack, J., Jones, D. and Ross, D. (1990), The Machine that Changed the World. Rawson and Association, New York.

Wright, G., Cairns, G. and Goodwin, P. (2009) Teaching scenario planning: Lessons from practice in academe and business. *European Journal of Operational Research*, Vol. 194, Issue 1, pp. 323–335.

Wu, S. (2006) The Quality of Design Team Factors on Software Effort Estimation. *Proceedings of IEEE International Conference* on *Service Operations and Logistics, and Informatics, SOLI '06*, 21st-23rd June, 2006, Faculty of Business, University of Macau, pp. 6 – 11.

Yetman, L. (2004) Project Management: Careful Planning or Crystal Ball?. *The CHAOS Report by the Standish Group, QAI 4th Quarter Journal*, pp. 8-11.

Yilmaz, M. and Chatterjee, S. (1997) Deming and the Quality of Software Development. *Business Horizons*, Vol. 40, Issue 6, pp. 51-58.

Zhang, H., Kitchenham, B. and Jeffery, R. (2007) Planning Software Project Success with Semi-Quantitative Reasoning. *Proceeding of the 2007 Australian Software Engineering Conference (ASWEC'07), IEEE*, 10th-13th April, 2007, University of North Wales, pp. 1-10.

Zowghi, D. and Nurmuliani, N. (2002) A Study of the Impact of Requirements Volatility on Software Project Performance. *Proceedings of the 9th Asia-Pacific Software Engineering Conference, IEEE*, 28th February- 3rd March, 2002, University of Technol, Sydney, NSW, Australia, pp. 1-9.

Zultner, R. (1993) TQM Technical Teams. *Communications of the ACM*, Vol. 36, no.10, pp. 79-91.

Zwikael, O. (2009) The Relative Importance of the PMBOK Guide's Nine Knowledge Areas during Project Planning. *Project Management Journal*, Vol. 40, no.4, pp. 94-103.

Zwikael, O. and Globerson, S. (2006) Benchmarking of Project Planning and Success in Selected Industries. *Benchmarking: An International Journal*, Vol. 13, no. 6, pp. 688-700.

Bibliography:

Agarwal, N. and Rathod, U. (2006) Defining Success for Software Projects: An Exploratory Revelation. *International Journal of Project Management*, Vol. 24, Issue 4, pp. 358-370.

Ahmad, N. and Laplante, P. (2006) Software Project Management Tools: Making a Practical Decision Using AHP. *Proceedings of the 30th Annual IEEE/NASA Software Engineering Workshop, IEEE*, April 2006, pp. 76-84, ISSN: 1550-6215.

Ahmad, S. (2002) Service Failures and Customer Defection: a Closer Look at Online Shopping Experiences. *Managing Service Quality*, Vol. 12, no. 1, pp. 19-29.

Aladwani, A. (2002) IT Project Uncertainty, Planning and Success: an Empirical Investigation from Kuwait. *Information Technology & People*, Vol. 15, no. 3, pp. 210-226.

Chahar, K., Taaffe, K., (2009) Risk Adverse Demand Selection with All-or-Nothing Orders. *Omega*, Vol. 37, Issue 5, pp. 996-1006.

Chang, C., Jiang, H., Di, Y., Zhu, D. and Ge, Y. (2008) Time-line Based Model for Software Project Scheduling with Genetic Algorithms. *Information and Software Technology*, Vol. 50, Issue 11, pp. 1142-1154.

Cheung, Y., Willis, R. and Milne, B. (1999) Software Benchmarks using Function Point Analysis. *Benchmarking: an International Journal*, Vol. 6, no. 3, pp. 269-279.

Clements, P., Jones, L., Nothrop, L. and McGregor, J. (2005) Project Management in a Software Product Line Organisation. *Software IEEE*, Vol. 22, Issue 5, pp. 54-62.

Dale, B., Wiele, A. and Williams, A. (2001) Quality- Why do Organisations Still Continue to Get it Wrong?. *Managing Service Quality*, Vol. 11, no. 4, pp. 241-248.

Davis, A. (1993) Software Requirements: Objects, Functions and States. Prentice Hall.

Day, R. (1993) Quality Function Deployment: Linking a Company with Its Customers. *ASQ Quality Press*, Milwaukee, Wisconsin.

Deng, S. and Xu, L. (2009) Mean-Risk Efficient Portfolio Analysis of Demand Response and Supply Resources. *Energy*, Vol. 34, Issue 10, pp. 1523-1529.

Duncan, R. (1972) Characteristics of Perceived Environments and Perceived Environmental Uncertainty. *Administrative Science Quarterly*, Vol. 17, no., 3, pp. 313-327.

Enkel, E., Kausch, C. and Gassmann, O. (2005) Managing the Risk of Customer Integration. *European Management Journal*, Vol. 23, Issue 2, pp. 203-213.

Enriquez, F., Osuna, A. and Bosch, V. (2004) Prioritising Customer Needs at Spectator Events. *International Journal of Quality & Reliability Management,* Vol. 21, no. 9, pp. 984-990.

Galasso, F. and Thierry, C. (2009) Design of Cooperative Processes in a Customer-Supplier Relationship: an Approach Based on Simulation and Decision Theory. *Engineering Applications of Artificial Intelligence*, Vol. 22, Issue 6, pp. 865-881.

Gautam, N., Singh, N., (2008) Lean Product Development: Maximising the Customer Perceived Value through Design Change (Redesign). *International Journal of Production Economics*, Vol. 114, Issue 1, pp. 313-332.

Genuchten, M. (1991) Why is Software Late? An Empirical Study of Reasons for Delay in Software Development. *IEEE Transactions on Software Engineering*, Vol. 17, no. 6, pp. 582-590.

Gil, I., Berenguer, G. and Cervera, A. (2008) The Roles of Service Encounters, Service Value, and Job Satisfaction in Achieving Customer Satisfaction in Business Relationships. *Industrial Marketing Management*, Vol. 37, Issue 8, pp. 921-939.

Githens, G. (2002) How to Assess and Manage Risk in NPD Programs: A Team-Based Risk Approach. *The PDMA Tool book for new Product Development*, John Wiley and Sons Inc.

Gonsalves, T. and Itoh, K. (2010) Multi-Objective Optimisation for Software Development Projects. *Proceedings of the International Multi Conference of Engineers and Computer Scientists (IMECS)*, 17th-19th March, 2010, Hong Kong, Vol. I, pp. 17-19.

Hahn, E. (2008) Production, Manufacturing and Logistics- Mixture Densities for Project Management Activity Times: A Robust Approach to PERT. *European Journal of Operational Research,* Vol. 188, Issue 2, pp. 450-459.

Hughes, B. and Cotterell, M. (2002) Software Project Management. The McGraw-Hill Companies, 3rd Edition.

Hybert, P. (1996) Five Ways to Improve the Contracting Process. *Quality Progress*, Vol. 29, Issue 2, pp. 65-70.

Jiang, J., Klein, G., Hwang, H., Huang, J. and Hung, S., (2004) An Exploration of the Relationship between Software Development Process Maturity and Project Performance. *Information & Management,* Vol. 41, Issue 3, pp. 279-288.

Kerzner, H. (2006) Project Management: A Systems Approach to Planning, Scheduling, and Controlling. John Wiley & Sons Inc., Hoboken, New Jersey, 9th Edition.

Levis, A. and Pagageorgiou, L. (2005) Customer Demand Forecasting via Support Vector Regression Analysis. *Chemical Engineering Research and Design*, Vol. 83, Issue 8, pp. 1009-1018.

Lewis, M. (2003) Cause, Consequence and Control: Towards a Theoretical and Practical Model of Operational Risk. *Journal of Operations Management*, Vol. 21, Issue 2, pp. 205-224.

Lin, C., Abdel-Hamid, T. and Sherif, J. (1997) Software-Engineering Process Simulation Model (SEPS). *Journal of Systems Software*, Vol. 38, Issue 3, pp. 263-277.

Liu, X. (2000) Software Quality Function Deployment. *IEEE Potentials*, Vol. 19, no. 5, pp. 14-16.

Lu, M., Madu, C., Kuei, C. and Winokur, D. (1994) Integrating QFD, AHP, and Benchmarking in Strategic Marketing. *Journal of Business & Industrial Marketing*, Vol. 9, no. 1, pp. 41-50.

Lu, X. and Ge, Y. (2003) Risk Analysis in Project of Software Development. *Proceedings the Engineering Management Conference of Managing Technologically Driven Organisations: the Human Side of Innovation and Change,* 2nd-4th Nov, 2003, Zhejiang University, Hangzhou, China, pp. 72-75.

Lui, F., Noguchi, K., Dhungana, A., Srirangam, V. and Inuganti, P., (2006) A Quantitative Approach for Setting Technical Targets Based on Impact Analysis in Software Quality Function Deployment (SQFD). *Software Quality Journal*, Vol. 14, no. 2, pp. 113-134.

Lukas, J. (2002) It Works! Risk Management on an IS Project. *Proceedings of the Project Management Institute Annual Seminars and Symposium,* 3rd-10th Oct, 2002, San Antonio, Texas, USA, pp. 23-35.

Maguire, S. (2002) Identifying Risks during Information System Development: Managing the Process. *Information Management & Computer Security*, Vol. 3, no. 3, pp. 126-134.

Mahanti, R. and Antony, J. (2005) Confluence of Six Sigma, Simulation and Software Development. *Managerial Auditing Journal*, Vol. 20, no. 7, pp. 739-762.

Matzler, K. and Hinterhuber, H. (1998) How to Make Product Development Projects More Successful by Integrating Kano's Model of Customer Satisfaction into Quality Function Deployment. *Technovation*, Vol. 18, no. 1, pp. 25-38.

Mauerkirchner, M. (1997) Event Based Modelling and Control of Software Development Processes. *Proceedings of International Conference and Workshop on Engineering of Computer-Based Systems, IEEE Transaction*, 24th-28th March, 1997, University of Upper Austria, Hagenberg, pp. 149-156.

Mefford, R., (2009) Increasing Productivity in Global Firms: The CEO Challenge. *Journal of International Management*, Vol. 15, Issue 3, pp. 262-272.

Appendix

List of Published Papers

Alkaabi, M., Khalil, R. and Stockton, D. (2010) Improving Operations Management Planning and Control of a Service Project with Lean Principles. *Proceedings of the Junior Scientist Conference*, 4th-7th April, 2010, Vienna, Austria, pp. 61-62.

Alkaabi, M., Khalil, R. and Stockton, D. (2009) Implementing Lean in Software Development Operations. *Proceedings of the International Conference on Product Lifecycle Management-PLM-SP5*, 6th-8th July, 2009, Bath University, Bath, UK, pp. 690-698.